BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
MASTER OF SCIENCE IN COMPUTER ENGINEERING

END-TO-END, REAL TIME AND ROBUST BEHAVIORAL PREDICTION MODULE WITH ROBOT OPERATING SYSTEM FOR AUTONOMOUS VEHICLES

BY

TOLGA KAYIN

MASTER OF SCIENCE THESIS

ANKARA - 2023

BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
MASTER OF SCIENCE IN COMPUTER ENGINEERING

END-TO-END, REAL TIME AND ROBUST BEHAVIORAL PREDICTION MODULE WITH ROBOT OPERATING SYSTEM FOR AUTONOMOUS VEHICLES

BY

TOLGA KAYIN

MASTER OF SCIENCE THESIS

ADVISOR

ASST. PROF. DR. ÇAĞATAY BERKE ERDAŞ

ANKARA - 2023

BAŞKENT UNIVERSITY

INSTITUTE OF SCIENCE AND ENGINEERING

This study, which was prepared by Tolga KAYIN, for the program of Master of Science in Computer Engineering, has been approved in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Computer Engineering Department by following committee.

Date of Thesis Defense: 11 / 10 / 2023

Thesis Title: End-to-End, Real Time and Robust Behavioral Prediction Module With Robot Operating System For Autonomous Vehicles

Examining Committee Members	**Signature**
Asst. Prof. Dr. Çağatay Berke ERDAŞ - Başkent University	……...
Prof. Dr. Uğur Murat LELOĞLU – University of Turkish Aeronautical Association	..……...
Assoc. Prof. Dr. Emre SÜMER - Başkent University	……...

APPROVAL

Prof. Dr. Faruk ELALDI

Director, Institute of Science and Engineering

Date: … / … / 2023

BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
MASTER'S / DOCTORAL THESIS / PROJECT ORIGINALITY REPORT

Date: 11/10/ 2023

Student's Name, Surname: Tolga KAYIN

Student ID : 22110457

Department: Computer Engineering

Program : Master of Science in Computer Engineering with Thesis

Advisor Title/Name, Surname: Dr. Asst. Prof. Çağatay Berke ERDAŞ

Thesis Title: End-to-End, Real Time and Robust Behavioral Prediction Module With

Robot Operating System For Autonomous Vehicles

The originality report of my master's thesis, as titled above, which consists of Introduction, Main Chapters, and Conclusion sections, totaling 46 pages, was generated by applying the following filters via the Turnitin plagiarism detection system by my thesis advisor on 11/10/2023. According to this originality report, the similarity rate of my thesis is 14%. The applied filters were:

1. Excluding the bibliography
2. Excluding quotations
3. Excluding text segments with less than five (5) overlapping words

I have reviewed the "Procedures and Principles for Obtaining and Using the Originality Report for Thesis Studies at Başkent University." I declare that my thesis conforms to the maximum similarity rates stipulated in the application rules, and I accept any legal responsibility that may arise if it is determined otherwise. I also affirm that the information provided above is accurate.

Student's Signature:

APPROVAL

Date: 11 /10 / 2023

Dr. Asst. Prof. Çağatay Berke ERDAŞ

Advisor's Signature:

ACKNOWLEDGEMENTS

I would like to thank to my advisor Asst. Prof. Dr. Çağatay Berke ERDAŞ for his advices and guidance. His valuable advices helped me during the whole period of my research.

I would also like to thank my family for supporting me throughout writing this thesis and my studies.

ABSTRACT

Tolga KAYIN

END-TO-END, REAL TIME AND ROBUST BEHAVIORAL PREDICTION MODULE WITH ROBOT OPERATING SYSTEM FOR AUTONOMOUS VEHICLES

Başkent University Institute of Science
The Department of Computer
Engineering 2023

In the world, where urbanization and population density are increasing, transportation methods are also diversifying and the use of unmanned vehicles is becoming widespread. In order for unmanned vehicles to perform their tasks autonomously, they need to be able to perceive their own position, the environment and predict the possible movements/routes of environmental factors, similar to living things. In autonomous vehicles, it is extremely important for the safety of the vehicle and the surrounding factors, to be able to forecast the probable future location of the objects around it with high performance so that the vehicle can plan itself correctly. Due to the stated reasons, the behavioral prediction module is a very important component for autonomous vehicles, especially in moving environments. In this study, a robotic behavioral prediction module has been developed to enable the autonomous vehicle to plan more safely and successfully. Data has been collected by driving with an autonomous vehicle, and the developed module has been tested. The relevant module has been integrated into the ongoing autonomy project. The proposed method has been observed to operate accurately and fast within up to three seconds.

KEYWORDS: Behavioral Prediction, Trajectory Prediction, Autonomous Vehicles, Robotic Operating System, ROS

Advisor: Asst. Prof. Dr. Çağatay Berke ERDAŞ, Başkent University, Department of Computer Engineering.

ÖZET

Tolga KAYIN

OTONOM ARAÇLAR İÇİN UÇTAN-UCA, GERÇEK ZAMANLI VE HATAYA DİRENÇLİ DAVRANIŞSAL TAHMİN MODÜLÜ

Başkent Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
2023

Küresel olarak nüfus yoğunluğunun arttığı şehirleşme sürecinde, ulaşım seçenekleri çeşitlenmekte ve insansız araçlar daha yaygın hale gelmektedir. İnsansız araçlar, kendi görevlerini otonom bir şekilde yerine getirebilmek için canlı organizmalar gibi çevrelerini algılayabilmeli, konumlarını belirleyebilmeli ve çevresel faktörlerin olası hareketlerini ya da yollarını tahmin edebilmelidir. Otonom araçlar, etkili bir planlama gerçekleştirebilmek için çevredeki nesnelerin gelecekteki pozisyonlarını doğru bir şekilde tahmin edebilmelidir. Bu, hem aracın güvenliği hem de çevredeki faktörlerin güvenliği açısından son derece kritik bir unsurdur. Davranışsal tahmin yeteneği olmayan bir otonom araç, tüm nesneleri sabit olarak varsayarak planlama yapar, ancak bu, otoyol koşulları veya şehir içi trafik senaryolarında araçların veya yayaların potansiyel yollarını hesaba katmadığında kazaların kaçınılmaz olduğu anlamına gelir. Bu çalışmada, güvenlik risklerini en aza indirmek amacıyla hızlı ve etkili bir robotik davranışsal tahmin modülü geliştirilmiştir. Otonom araç ile sürüş yapılarak veri toplanmıştır ve geliştirilen modül test edilmiştir. İlgili modül hali hazırdaki çalışılan otonomi projesine entegre edilmiştir. Önerilen metodun üç saniyeye kadar, belirli çevresel obje sayısında, başarılı ve hızlı bir şekilde çalıştığı görülmüştür.

ANAHTAR KELİMELER: Davranışsal Tahminleme, Güzergah Tahminleme, Otonom Araçlar, Robot İşletim Sistemi

Danışman: Dr. Öğr. Üyesi Çağatay Berke ERDAŞ, Başkent Üniversitesi, Bilgisayar Mühendisliği Bölümü.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF SYMBOLS AND ABBREVIATIONS

ADE	Average Displacement Error
ANN	Artificial Neural Networks
AV	Autonomous Vehicle
CNN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short-term Memory
CSAA	Constant-Steering Angle & Acceleration
CSAV	Constant-Steering Angle & Velocity
CTRA	Constant-Turn Rate & Acceleration
CTRL	Constant-Turn Rate & Velocity
CV	Constant Velocity
CVAE	Conditional Variational Auto-Encoder
DBN	Dynamic Bayesian Network
D-IRL	Deep Inverse Reinforcement Learning
DNN	Deep Neural Network
DP	Dirichlet Process
FDE	Final Displacement Error
KNN	K-Nearest Neighbors
GAIL	Generative Adversarial Imitation Learning
GAN	Generative Adversarial Network
GCN	Graph Convolutional Network
GAT	Graph Attention Network
GP	Gaussian Process
GMM	Gaussian Mixture Model
GNN	Graph Neural Network
GPS	Global Positioning System
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
HMTP*	Hidden Markov model-based Trajectory Prediction Star
IMM	Interacting Multiple Model
IMU	Inertia Measurement Unit
IRL	Inverse Reinforcement Learning
KF	Kalman Filtering
KNN	K-Nearest Neighbors
LSTM	Long Short-term Memory
ML	Machine Learning
MPC	Model Predictive Control
MR	Miss Rate
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROS	Robotic Operating System
ROS2	Robotic Operating System 2
SKF	Switched Kalman filter
ST-LSTM	Spatio-Temporal Long Short-term Memory
SVM	Support Vector Machine
VOI	Value of Information
ZMQ	ZeroMQ

1. INTRODUCTION

The function and importance of autonomous vehicles are increasing day by day. It is foreseen that autonomous vehicles will play an important role in the future in order to reduce the density of transportation and to eliminate human-induced accidents. Apart from transportation, autonomous vehicles are becoming more and more common in areas such as agriculture, health and education.

Autonomous vehicles are inspired by living things; they consist of modules such as perception to detect the environment, localization to determine its own position, planning to where and how to go, control for its movement and behavioral prediction for possible movement routes of surrounding objects. Middleware such as Robotic Operating System (ROS)_[1], ZeroMQ (ZMQ)_[2] and Robotic Operating System 2 (ROS2)_[3] are needed for these modules to communicate with each other correctly and completely. These middlewares enable modules to transmit the desired message to the relevant module. Thanks to the ROS middleware tools that are used in the study, it also provides benefits such as visualizing, recording and observing data.

1.1 Problem Definition

In autonomous vehicles consisting of modules such as control, canbus, perception, localization, planning and behavioral prediction, it is extremely important to predict the future position of the objects around it with high performance for the vehicle to plan correctly. The module that predicts the possible routes of objects in the environment of autonomous vehicles is the behavioral prediction module. Thus, the behavioral prediction module is one of the most important factors for the accurate result of the planning module in autonomous vehicles. The behavioral prediction module generates output that predicts future positions by keeping the past positions of objects around the ego vehicle. This output creates an input to the planning module by combining the objects found by the detection module. An autonomous vehicle without a behavioral prediction module will consider all objects as static and plan accordingly, but in highway conditions or urban traffic scenarios, an accident will be inevitable if the possible routes of vehicles or pedestrians are not taken into account. To give an example from scenarios that are frequently experienced in daily life, in order for an autonomous vehicle to consider a pedestrian preparing to cross the street, the autonomous vehicle must know the pedestrian's possible route. Similarly, while the autonomous vehicle is changing lanes, it must calculate the possible route according to the speed of the vehicle

from behind, otherwise there will most likely be an accident. In this thesis, the problem of finding the possible routes of moving objects around the autonomous vehicle was focused on.

1.2 Problem Solution

A behavioral prediction module has been developed to solve the problem specified in Section 1.1. The possible routes of the objects around the autonomous vehicle were estimated so that the planning module could draw a path by taking the possible routes of moving objects into account. Developing a fast and successful behavioral prediction module in order to prevent the mentioned risks will make significant contributions to both the literature and life.

The developed behavioral prediction module is based on ROS and works in real-time. Features such as ROS middleware, dynamic history hold and release structure, direction error correction, covariance distribution visualization, and message type matching suitable for planning have been added to the multi modal Conditional Variational Auto Encoder based (CVAE-based) model_[4]. Thus, an end-to-end autonomy module structure was created that sends the possible routes of the surrounding vehicles to the planning module.

1.3 Aims & Objectives

With the behavioral prediction module to be obtained as a result of this thesis, the possibility of an autonomous vehicle making a mistake in the environment of moving objects will be significantly reduced. Apart from contributions such as writing real time inference to the current model, adding ROS infrastructure, and correcting or filtering erroneous data; the addition of Convolutional Long Short-term Memory (ConvLSTM) to a GNN-based model and the acceleration of this model by using TensorRT are the most important contributions of the thesis to the literature.

1.4 Organization of the Thesis

In the next part of the study, behavioral prediction approaches and studies in the literature review section will be summarized, then information about the methodology used in the study will be given and the developed module will be explained in detail in the material and methods section. Afterward, in the results section, the results obtained with the test data will be shared and in the findings section test results and behavioral prediction module will be discussed. Finally, the conclusion of the study will be expressed and deductions about the test results will be shared in the conclusion section.

2. LITERATURE REVIEW

2.1 Previous Literature

In the literature, there is very little research in the field of behavioral prediction compared to areas such as perception, localization, and planning of autonomous vehicles. The biggest reason for this is that it is more difficult to determine the location of environmental factors in the future than the problems in other areas. When the trajectory prediction approaches are examined [5][6], although there are approaches such as representation, contextual factors, modeling, situational awareness, the modeling approach will be used as the main approach in categorizing the studies in this article. In addition, information will be provided in terms of representation, output and situational awareness types for the studies. When the studies are examined in terms of modeling methods; behavioral prediction methods are shown in Figure 2.1, they consist of physics-based, machine learning based, deep learning-based and reinforcement learning-based methods.

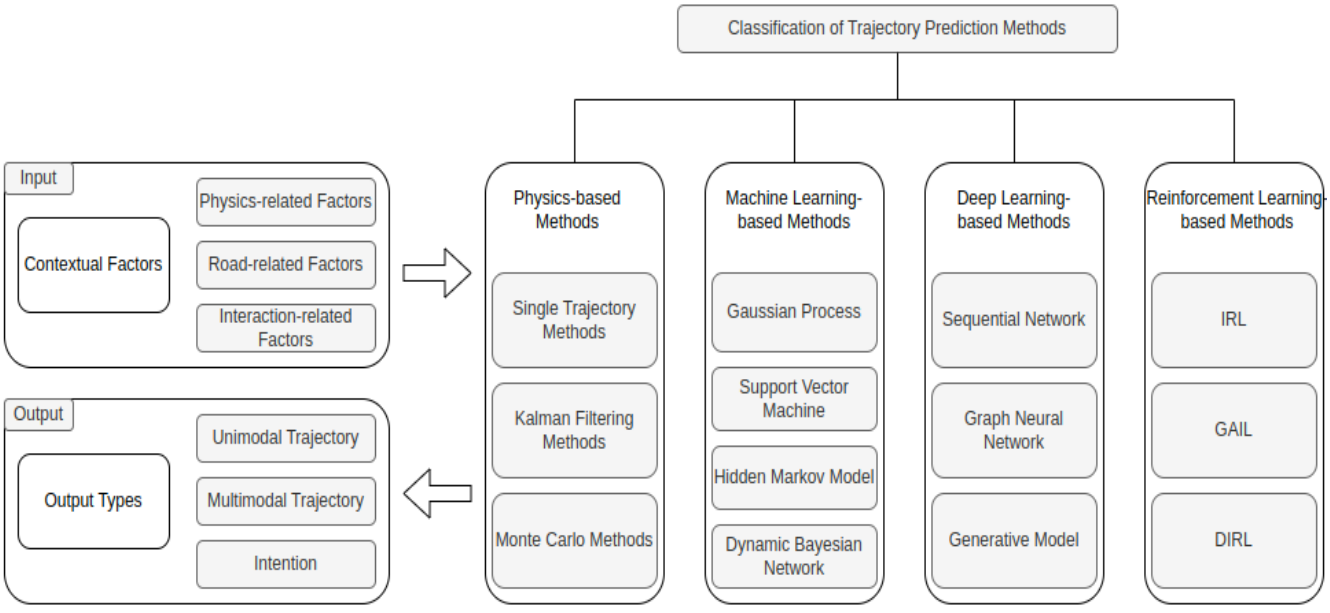

Figure 2.1 Trajectory Prediction Methods

Physics-based methods take information from the dynamics and kinematics of the vehicle. They consist of single-trajectory, Kalman filter (KF) and Monte Carlo methods.

Models using a single trajectory mostly use the kinematic information of the vehicle. Although there are also models that use the dynamic information of the vehicle, these models are more complex. Dynamic models consider all forces that govern motion. Dynamic models are highly complex due to the factors involved. For example, for a vehicle, the dynamic model considers the forces acting on the tires, the driver's actions and their effects on the vehicle's engine and transmission. For trajectory prediction, it does not make much sense to use a dynamic model to model such complex behavior unless you intend to run a control-oriented application [7]. Kinematic models are more commonly used than dynamic models due to their simpler structure. One of the most commonly used is Constant Velocity (CV). A simple example of a kinematic model is the CV model used in [8]. The CV model assumes that the recent relative motion of an object determines its future trajectory. Similarly, Ammoun et al. [9] and Schubert et al. [10] estimated the possible trajectories of the vehicle using the Constant Acceleration (CA) method. The CA method estimates the future acceleration of the vehicle from the past acceleration data, these acceleration estimates are converted into position information and the possible position of the vehicle is found. Lytrivis et al. [11] used Constant Turn Rate and Velocity (CTRL) and Constant Turn Rate & Acceleration (CTRA) models and Batz et al. [18] used Constant Steering Angle & Velocity (CSAV) and Constant Steering Angle & Acceleration (CSAA) models by adding wheel data to the model.

The single trajectory methods given as an example [7]-[11] use non-noise data from the vehicle. In contrast, the Kalman Filtering method can handle the uncertainty of present vehicle conditions or noise such that the noise and its physical model are modeled by a Gaussian distribution. The prediction and adjustment phases are integrated within a continuous loop. The vehicle state mean and covariance matrices are determined at each future time step and computed as mean trajectories with associated uncertainties. In their work, Kempchen and colleagues [12] introduce an Interacting Multiple Model (IMM) for generating multiple possible trajectories. The Switched Kalman Filter (SKF) [13], on the other hand, relies on a series of Kalman filter techniques to model the vehicle's physical behavior and transition between these models.

Apart from the Single Trajectory and Kalman filter method; there is the Monte Carlo approach, which can imitate state distributions approximately. It involves random testing of

input variables and utilizes a physical model to generate potential future trajectories. In their work, Okamot and colleagues [14] introduce a model based on maneuvers, employing Monte Carlo methods to anticipate future trajectories based on identified maneuvers. Similarly, Weng and their team [15] forecast trajectories using the Monte Carlo approach and optimize the reference trajectory using MPC.

While the physics-based methods described above [7]-[15] kinematics and dynamics were used as inputs, road-related factors were also used as inputs in Coelingh et al. [16] and Xie et al. [17] studies. These studies mostly output unimodal trajectory. There are also studies that outputs multimodal trajectory or intention as in Hermes et al. [19].

Unlike physics-based methods, machine learning methods are based on the principle of obtaining predicted trajectories by data mining. On the other hand, learning-based models tend to capture and incorporate changes caused by long-term dependencies and external factors, compared to physics-based models that are limited to low-level motion characteristics and are poor at estimating long-term motion dependencies. The most widely used machine learning methods are Decision Tree, Hidden Markov Model (HMM), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Dynamic Bayesian Network (DBN) and Gaussian Process (GP) methods.

When applying GP to predict a trajectory, the trajectory is considered as GP samples tested along the time axis. A sample is symbolized by N discrete points for mapping into N-dimensional space. The samples then fill an N-proportional Gaussian distribution in N-proportional area. GP could also be applied to model interaction-related components, Trutman et al. [20] use GP to avoid joint collisions and solve frozen robot problems. GP and Dirichlet process (DP) are applied to determine the motion process, and a non-parametric Bayesian network is applied to extricate potential movement models by Guo et al. [21].

Kumar and colleagues [22] suggested that as Support Vector Machines (SVM) are capable of providing classification probability attributes, they proposed a multi-layer architectural approach that combines SVM with Bayesian filtering to recognize lane change maneuvers and get more error-free identification results.

In real life, only visible states can be observed on vehicles, but we cannot intuitively express the hidden states. Hence, there is a requirement to create a Markov process that incorporates concealed states and identify the inherent state of an event through a collection of observable states associated with the likelihood of the concealed state. This concept is

known as the hidden Markov model (HMM). Building upon the HMM framework, Qioa and their team [23] introduce an algorithm called HMTP, which dynamically selects parameters to replicate real-world scenarios characterized by variable speeds. In [24], their HMM connection with fuzzy logic is applied to predict driver maneuvers. The author of [25] presents a DBN representing driver behavior and vehicle trajectory. DBNs have Markov properties. We can extend the state with more information to satisfy the Markov assumption. In [25], this is done by adding all relevant information about the process to the DBN in the form of a vector.

Although the outputs of the studies in this method are mostly multimodal, it has been observed that the model's performance increases as the situational awareness states such as map-aware, scene-aware and interaction-aware increase.

One of the situational awareness scenarios in the field of behavior prediction is "map aware," where an HdMap is provided as input to the model, annotated with roadways, lanes, traffic signs, and similar elements. HdMap, short for High-Definition Map, is a specialized type of digital map that provides highly detailed and accurate information about the road network, traffic infrastructure, and surrounding environment. "Scene-aware" involves incorporating the objects in the vicinity and areas on the map, such as intersections and road junctions, into the model together. On the other hand, "interaction-aware" entails including the interactions between objects in the environment in the model along with the map as input.

Deep-learning based methods are based on the principle of obtaining a predicted trajectory as a result of the model obtained by performing various feature extraction and regression operations of the historical trajectory. Most of the studies in the field of trajectory prediction consist of deep learning-based approaches. There are more than a hundred studies based on deep learning. Only mainstream studies are considered in this study.

Deep-learning based methods consist of sequential networks, graph neural network (GNN) and Generative Model methods. Sequential network methods consist of Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), RNN and CNN and Attention Mechanism, while Generative Model methods consist of Generative Adversarial Network and Conditional Variational Auto Encoder methods.

One of the most popular studies with RNN & CNN is DESIRE [26], whose goal is to predict the future positions of multiple interacting agents in a dynamic (driving) scene. This takes into account the multimodal nature of future projections. For example, even in the same

situation, the future can be different. It can predict potential future outcomes and make strategic predictions based on them, making inferences based not only on past movement history but also on scene context and agent interactions.

Another example of work with a sequential network is [27] the modified version of LSTM i.e. ST-LSTM (Spatio-temporal LSTM) is used in [27] where the interaction of multiple vehicles and its effect on the trajectory of Value of Information (VOI) is estimated.

When the studies conducted with GNN are examined, Deal and colleagues [28] employ two widely recognized graph networks: Graph Convolutional Networks (GCN) and Graph Attention Networks (GAT) for predicting trajectories based on interaction-related elements and validate their effectiveness. To explain further, GCN's main concept revolves around learning a mapping function that can extract features recognizing interactions from a node's features within the figure and those of its neighboring nodes. On the other hand, GAT utilizes an attention mechanism to determine the weights between nodes when combining feature data. Another instance of Graph Neural Networks (GNN) is presented by Li and team [29] in their GRIP model, which utilizes both static and dynamic graph networks to forecast the trajectories of road users. Notably, in the latter part of 2019, GRIP achieved the top position in the Baidu Apolloscape dataset [30]. It's worth mentioning that previous iterations of GRIP employed the LSTM encoder/decoder, whereas the current version uses the GRU for both encoding and decoding.

Another deep learning approach is the generative model approach. Trajectory prediction generative models include Generative Adversarial Networks (GANs) and Conditional Variational Autoencoders (CVAEs). Gupta and colleagues [31] employ a Generative Adversarial Network (GAN) known as SGAN for predicting pedestrian paths. The generator in SGAN utilizes an LSTM encoder, a pooling module, and an LSTM decoder to generate estimated trajectories, while the discriminator employs LSTMs to assess the quality of the predicted trajectories. On a related note, Sohn and their research team [32] introduce a method involving conditional Variational Autoencoder (CVAE) to address structured prediction tasks. When it comes to predicting trajectories, combining variations of CVAE and RNN in the roles of encoder and decoder proves to be an effective approach for trajectory generation.

Deep learning-based studies can provide more comprehensive output and input compared to physics and machine learning-based studies. These studies mostly take interaction-aware inputs and provide multimodal or intention type output.

The reinforcement learning approach, which has been extensively studied in recent years, also appears in predicting trajectory. The reinforcement learning method is based on the decision-reward principle, focusing on finding the decision that will maximize the reward.

The Reinforcement Learning method consists of Inverse Reinforcement Learning(IRL), Generative Adversarial Imitation Learning(GAIL) and Deep Inverse Reinforcement Learning(D-IRL) methods.

Based on research using these methods; In Sun et al. [33] study, interaction related elements are taken into account to achieve probabilistic estimation for AVs by using IR. Future trajectory distribution is defined by driving manoeuvers. Kufler et al. [34] extended GAIL to their RNN optimization to show the behavior of a human driver, discriminators evaluate steps and actions. Choi et al. [35] combine the partially observable Markov decision process (POMDP) within the GAIL framework and propose a method to train the model using the discriminator reward function. The prediction problem is nonlinear, so nonlinear mapping should be used for generalizable function approximation. Wulfmeier et al. [36] propose a deep inverse reinforcement learning (DIRL) framework for approximating complex nonlinear reward functions. Some D-IRL approaches get history tracks as input. Considering driving characteristics and route shape, the researchers in reference [37] began by applying reinforcement learning (RL) to create a Markov Decision Process (MDP). They then acquired the best driving strategy through Inverse Reinforcement Learning (IRL) and employed a Deep Neural Network (DNN) to formulate a reward system. In a separate work, Jung and colleagues [38] introduced a convolutional Long Short-Term Memory (LSTM) approach to extract feature representations from their LIDAR and trajectory data, considering factors like inertia, the environment, and societal influences. These extracted features are combined with the resulting reward map to predict the traversability map.

Reinforcement learning methods, similar to deep learning methods, can take extensive inputs such as road and scene related factors and provide comprehensive outputs in the form of unimodal and intention.

When Table 2.1 and Table 2.2 are examined, although physics-based and machine-learning-based methods require low computational load, their accuracy notably decreases, especially after 2 seconds. Compared to these two methods, deep learning and reinforcement-based learning methods can predict longer time successfully, although they overlay more computation load. When deep learning and reinforcement-based methods are compared, it is seen that the deep learning-based method is more successful.

2.2 Limitations of Previous Research

Previous researches are basically based on model studies. Studies on the integration of models with planning, detection and localization modules are limited. The majority of models are concentrated on predicting either vehicle-only or pedestrian-only trajectories. In many research articles, comparisons and tests have been made on displacement metrics, but there is no information about the working speed of the algorithms.

As seen in Table 2.2, some studies give multi trajectory and some single outputs. Integration of multi-trajectory output algorithms with planning algorithms is more complex. However, it is important to assign the correct cost values to the multi-trajectory outputs for the system to give accurate output.

An open source, ROS structured, end to end, configurable, robust, multi-class behavioral prediction module could not be found. The most related work found [39] is one that predicts pedestrian-only trajectories with ROS.

Table 2.1 Comparison of the trajectory prediction RMSE results of models using various methods trained on NGSIM dataset under highway condition [5]

Classification Methods	***Models***	***RMSE(m)***				
		1s	***2s***	***3s***	***4s***	***5s***
Single Trajectory	Constant Velocity[40]	0.73	1.78	3.13	4.78	6.68
Kalman Filtering	IMM-KF[41]	0.58	1.36	2.28	3.37	4.55
HMM	C-VGMM+VIM[42]	0.66	1.56	2.75	4.24	5.99
RNN	M-LSTM[43]	0.58	1.26	2.12	3.24	4.66
RNN	MFP-1[44]	0.54	1.16	1.90	2.78	3.83
CNN and RNN	CS-LSTM(M)[45]	0.62	1.29	2.13	3.20	4.52
Attention Mechanism	MHA-LSTM[46]	0.41	1.01	1.74	2.67	3.83
GNN	GRIP++[29]	0.38	0.89	1.45	**2.14**	**2.94**
GNN	GISNet[47]	**0.33**	**0.83**	**1.42**	2.14	3.23
Generative Model	MATF-GAN[48]	0.66	1.34	2.08	2.97	4.13
Generative Model	TS-GAN[49]	0.60	1.24	1.95	2.78	3.72
IRL	L-IRL[50]	1.12	2.29	2.31	3.38	4.45
GAIL	GAIL-GRU[51]	0.69	1.51	2.55	3.65	4.71
DIRL	MEDIRL[52]	1.35	2.57	2.83	3.69	4.88
DIRL	DN-IRL[53]	0.54	1.02	1.91	2.43	3.76

Table 2.2 Comparison of the trajectory prediction FDE, ADE and MR results of models using various methods trained in Argoverse data set under urban condition [5]

Classification Methods	***Models***	***K[1]=6***			***K[1]=1***		
		minFDE	***minADE***	***MR***	***minFDE***	***minADE***	***MR***
Physics-based	CV[54]	7.57	3.39	0.82	7.89	3.53	0.84
Machine Learning-based	NN+map[54]	4.03	2.08	0.58	8.12	3.65	0.84
RNN	LSTM+map[54]	5.44	2.34	0.69	6.81	2.96	0.81
RNN	Jean[55]	1.49	0.93	0.19	4.18	1.86	0.63
Attention Mechanism	SceneTransformer[56]	**1.23**	**0.80**	0.13	-	-	-
Attention Mechanism	mmTransformer[57]	1.34	0.84	0.15	-	-	-
GNN	LaneGCN[58]	1.36	0.87	0.16	3.78	1.71	0.59
GNN	DenseTNT[59]	1.45	0.93	**0.11**	-	-	-
GNN	LaneRCNN[60]	1.45	0.90	0.12	**3.69**	**1.69**	**0.57**
Generative Model	PRIME[61]	1.56	1.22	0.12	3.82	1.91	0.59

[1] output trajectory numbers

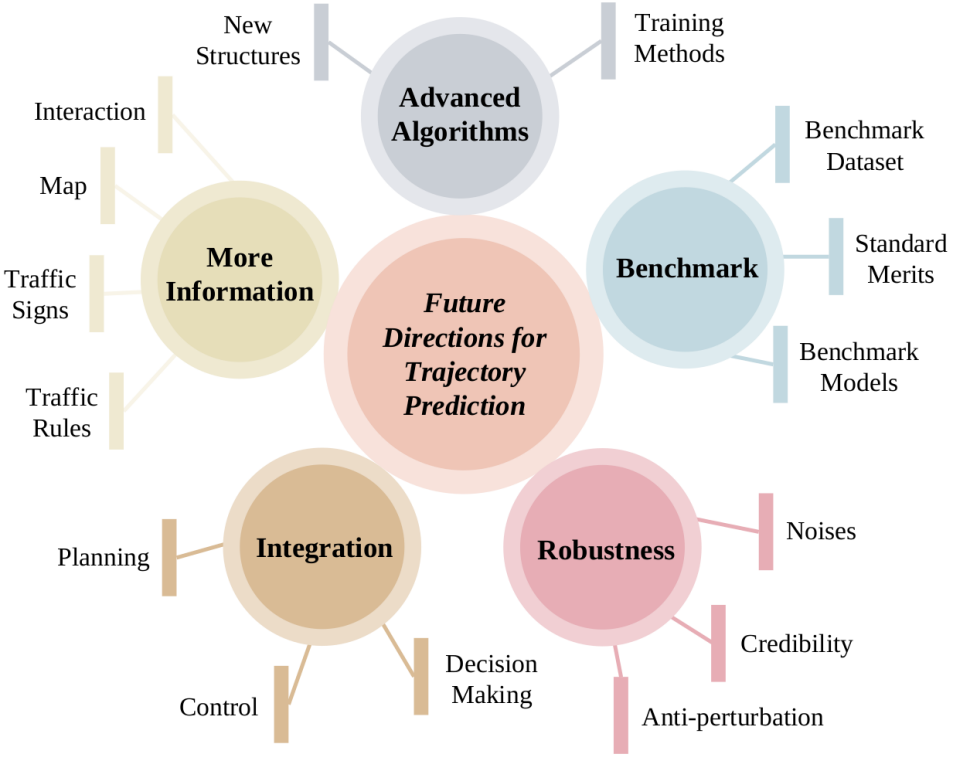

Figure 2.2 Behavioral Prediction research areas [5]

2.3 Contributions of the Study

In the thesis study, research has been conducted on the areas of behavioral prediction shown in Figure 2.2, as outlined below:

• Efforts have been made to accelerate the model using TensorRT, aiming to achieve advanced algorithms as a result.

• The developed module has been integrated with the planning module.

• Data has been collected for testing the algorithm and benchmarking it against other datasets.

• Robustness has been enhanced through the correction and filtering of data from the perception system.

• The model has been enriched by feeding it with additional heading data, providing more information to the model.

3. MATERIALS & METHODS

3.1 Datasets

Data sets are required for training or testing deep learning, machine learning and reinforcement-based behavioral prediction models. These data sets may consist of sensor data such as LIDAR, camera, radar, GPS, IMU, map data such as HDmap, vector map and their annotations, as well as perception or localization layer output for higher-level autonomy studies such as behavioral prediction. The main ones are KITTI [62], nuScenes [63], Argoverse [64] and NGSIM [65] datasets.

The model that is worked on in this thesis is trained on nuScenes dataset. NuScenes dataset is a large-scale autonomous driving dataset that has several distinct datasets such as nuPlan [66] for planning, nuScenes for perception, nuImages [67] for image level operations. For this purpose, 1,000 driving scenes were collected in Boston and Singapore, two cities known for heavy traffic and extremely challenging driving conditions. 20-second scenes are manually selected to depict a variety of interesting driving maneuvers, traffic situations, and unexpected behavior. The complexity of nuScenes encourages the development of methods that enable safe driving on urban roads with dozens of objects per scene. In addition, by collecting data on different continents, developers can work on generalizing computer vision algorithms for different locations, vegetation, vehicle types, road signs, weather conditions, left-hand and right-hand traffic.

To assist common computer vision tasks, such as object detection and tracking, it is annotated 23 object classes with accurate 3D bounding boxes at 2Hz over the entire dataset. Additionally, it is annotated object-level features such as visibility, movement and pose.

In March 2019, the full nuScenes was released dataset with all 1,000 scenes. The full dataset comprises approximately 1.4M camera images, 390k LIDAR sweeps, 1.4M RADAR sweeps and 1.4M object bounding boxes in 40k keyframes. However, new features (map layers, raw sensor data, etc.) are being added day by day. It is also organized the nuScenes 3D detection challenge as part of the Workshop on Autonomous Driving at CVPR 2019.

The nuScenes dataset leverages the KITTI dataset. nuScenes is the first large dataset to provide data from the entire autonomous vehicle sensor suite (6 cameras, 1 LIDAR, 5 radars, GPS, IMU). Compared to KITTI, nuScenes contains seven times more object annotations.

In order for the study to be tested, a Rosbag was collected with sizes 19.2 GB and 17.8 GB by 10- and 12-minute driving in the Mustafa Kemal district of Ankara/Turkey. As seen in Table 5.1 Rosbag data consists of the ego vehicle's localization output and perception output obtained by sensor fusion which includes positions, orientations, speeds, sizes of adjacent objects, transform information, camera images and visualization markers which include bounding box markers. In detail ego vehicle's localization output includes the ego vehicle's position, orientation and speed, perception output includes positions, orientations, speeds, sizes of adjacent objects, transform information includes relative positions and orientations of global map, local map and sensors, visualization markers includes bounding box markers of vehicles, pedestrians and unknown objects.

Dataset Type: Rosbag	
Size : 39 GB	**Duration: 13 min**
Data	***Data Description***
• Image Data	• Image data from camera
• Localization data	• Position and orientation data from localization module
• Camera Object Detections	• Object detections from camera
• Sensor fusion output	• Object detections from perception module
• Tf and Tf static	• Static and dynamic transformations of sensors, local and global map
• Lidar Data	• Pointcloud data from Lidar
• Visualization Markers	• Visualization Markers for 3D Worldmodel

Table 5.1 Collected Rosbag for testing

3.2 Evaluation Metrics

Reliable and generic metrics are needed to measure the success of the studies. Some metrics that can be used to compare related works are given below:

Root Mean Squared Error (RMSE): Root mean square error computes the square root of the mean squared forecast error. As shown in the equation (1) while $\hat{y}_i$ stands for estimated value (m), y_i indicates observed value and n represents the quantity of samples.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{1}$$

Average displacement error (ADE): The mean separation between the forecasted trajectory and the actual path. In the formulas (2) given below, x_i and y_i stand for predicted trajectory for one second interval in meters at x and y axes respectively, x_i^{GT} and y_i^{GT} indicate observed trajectory for one second interval in meters at x and y axes respectively, and T is time in seconds.

$$ADE = \frac{1}{T}\sum_{t=1}^{T} \sqrt{(x_i - x_i^{GT})^2 + (y_i - y_i^{GT})^2} \tag{2}$$

Final displacement error (FDE): As shown in the equation (3) FDE represents the gap between the ultimate prediction outcomes and the actual observed position. In the equation (3) given below, x_T and y_T stand for predicted trajectory for one second interval in meters at x and y axes respectively, x_T^{GT} and y_T^{GT} indicate observed trajectory for one second interval in meters at x and y axes respectively, and T is time in seconds.

$$\text{FDE} = \sqrt{(x_T - x_T^{GT})^2 + (y_T - y_T^{GT})^2} \tag{3}$$

Miss Rate (MR): Miss rate is the ratio of trajectories estimated outside 2 meters or more according to ground truth to all trajectory estimates. In the equation (4) given below, ‘misses’ are the forecasted paths located more than 2 meters away from the ground truth, while 'hits' are the predicted paths found within a 2-meter proximity of the ground truth.

$$\text{MR} = \frac{misses}{hits+misses} \tag{4}$$

3.3 Robotic Middleware

Robotics middleware refers to the intermediate software layer utilized within intricate control systems of robots. This kind of middleware is purposefully designed to manage the intricacies and diversity of both hardware and applications involved in robot control. Its role encompasses the seamless integration of emerging technologies, simplification of software architecture, concealing the intricacies of lower-level communication and sensor disparities, enhancement of software quality, reutilization of robot software infrastructure for various research endeavors, and reduction of production expenses.

It can be likened to "software glue," streamlining the path for robot developers to concentrate on their specific focal areas. Among the array of available middleware options, ROS (Robot Operating System) emerged as the preferred choice due to its extensive presence in existing literature and its incorporation within the ongoing project. ROS represents a compilation of software frameworks tailored for the creation of robotics software across heterogeneous computing clusters. It offers standard services akin to those provided by an operating system, including hardware abstraction, control over low-level devices, implementation of frequently utilized functions, inter-process communication, and management of software packages.

Among the array of available middleware options, ROS emerged as the preferred choice due to the following reasons;

- It is prevalent middleware in the literature.
- It provides bridge and communication between hardware and software
- It has peer to peer structure, system consists of separate nodes which is called Rosnode,

- As it is tool-based, has useful tools for visualization, diagnostics, data recording, logging, data plotting,
- It has a multi-lingual structure; Rosnodes can be written in C++, Python, MATLAB, Julia, Java etc.
- Due to Rosnodes are in simple structures, it is easy to modify and adapt to other structures
- ROS and its libraries are open source and expanding day by day.

3.4 Model Architecture

Trajectron++ has been used as the model in this thesis study for the following reasons;

- It is in a multi-classification structure that can give a trajectory output according to both vehicle and pedestrian.
- Compared to other datasets, the nuScenes dataset in which the model is trained is new and contains map data and interaction information of objects.
- As mentioned in the literature review section, many studies are showing that graph-structured recurrent models are more successful. However, the Trajectron++ model uses a CVAE in a graph structured neural network,
- It provides both unimodal and intent output, so that more and more flexible information can be provided to the planning module,
- It is an open-source model,
- It is a scene-aware model that takes into account the map and the interactions of other nodes with each other.

Trajectron++ is in a map and interaction-aware structure as shown in Figure 4.1. The model consists of a 2-part structure, encode and decode. In the encoding section, environmental objects such as vehicle pedestrians enter the LSTM first. Briefly, LSTM is a kind of recurrent neural network capable of learning long-term dependencies. In RNN output from the last step is fed as input in the current step. In general, RNNs have a vanishing gradient problem. The vanishing

gradient problem in RNNs refers to a challenge where the gradients (derivatives) used for updating the network's parameters during training become extremely small as they are propagated backward through time. This issue leads to the network's weights being updated very slowly, or not at all, which can result in poor convergence and slow learning. LSTM can overcome this problem with a forget gate, which is designed to pass information between memory cells to store the most important previous information.

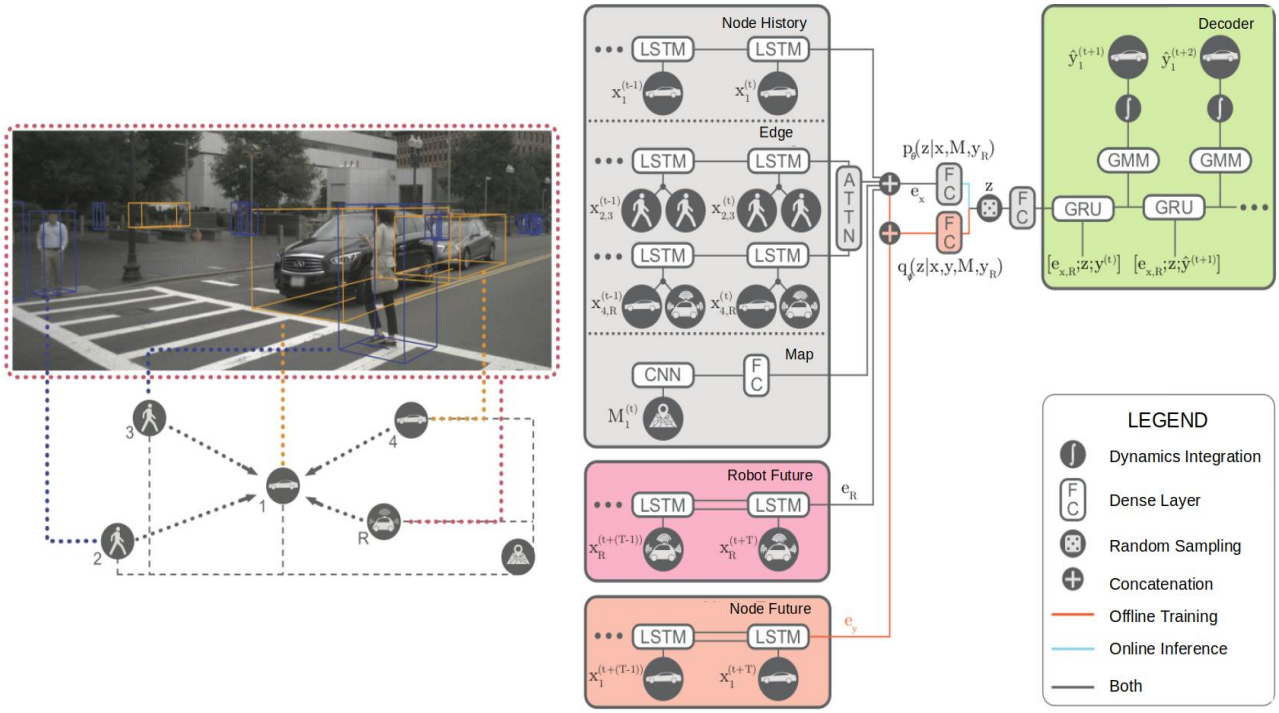

Figure 3.1 Trajectron++ model architecture

Edge parts enter the attention mechanism in addition to LSTM. The attention mechanism looks at an input sequence and decides at each step which other parts of the sequence are important. Map data enters the CNN model. CNN is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. CNN is a simple sequential architecture. The flight path history is used as input and is fed through a fully connected layer of fixed size. Convolutional layers are stacked and used to ensure temporal uniformity. Finally, the features of the last convolutional layer are combined and passed through a fully combined layer to generate all predicted locations simultaneously.

After the CNN layer, the map data enters the dense layer. The dense Layer is a simple layer of neurons in which each neuron receives input from all the neurons of the previous layer.

Dense layers are the most commonly used layers in models. In the background, the dense layer performs matrix-vector multiplication. The values used in the matrix are actually parameters that can be trained and updated using backpropagation. The output produced by the dense layer is an 'm' dimensional vector. So basically, dense layers are used to change the dimension of the vector. Dense layers also apply operations such as rotation, scaling, and translation to vectors.

Node history, edge history, map, robot future and node future information are entered into the decode section by concatenating and entering the dense layer. This information is decoded by going through the Gated Recurrent Unit (GRU), Gaussian Mixture Model (GMM) and Dynamic Integration stages, respectively.

GRUs are very similar to LSTM. Just like LSTM, GRU uses gates to control the flow of information. Unlike LSTM, it does not have a separate cell state. It only has a hidden state. GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem associated with standard recurrent neural networks. To solve the vanishing gradient problem in standard RNNs, the GRU uses so-called update and reset gates. Basically, these are two vectors that determine what information is passed to the output. What makes them special is that they can be trained to retain information long ago without obscuring it over time or removing information irrelevant to the prediction.

A GMM is a category of probabilistic models in which every data point generated is derived from a finite mixture of Gaussian distributions with no known parameters. Finally, dynamic integration means that the dynamic constraints of the vehicle or pedestrian are also taken into account. When some clusters may be ‘wider’ than others or clusters may overlap, GMM should be used. If one Gaussian model is used, it cannot handle the data set generated by multi-Gaussian models, so GMM is introduced -- use multi-Gaussian models and mix them into one with certain weights.

Studies on this model; ROS middleware has been added, the data from the perception module has been adapted to the input data format in the model. In addition, a dynamic history hold/drop structure has been created, so the history of tracked objects is accumulated, and the untracked object is prevented from entering the model. Heading from point cloud data and heading from position data are handled correctly so the behavior prediction module gets the accurate headings of surrounding objects.

3.5 The Developed Software Methodology

Due to the added ROS middleware, this software processes the incoming data from the perception subsystem, finds possible trajectories and sends them to the planning subsystem. In this way, the vehicle also considers the trajectories of its moving objects while planning.

The developed behavioral prediction module is shown in Figure 3.1 as a flowchart and pseudo code. First, by listening to the output of the perception module, information such as the position, class and orientation of the surrounding objects is obtained, and then dictionaries are created for the object at a certain speed and attention radius. As long as the ROS connection is open, the object information from the perception output is accumulated in the relevant dictionaries. This dictionary sizes have been chosen as twelve timesteps due to the detection system operating at 10fps, the common usage of a 2-second history in the literature, and the consistent results obtained in the conducted tests. When the objects reach enough history, they are converted into a data structure suitable for the model and entered as input to the model. 6-second predicted trajectories are output from the model. This estimated trajectory information is converted into the message types for the planning module and visualization. These messages are published to relevant modules by ROS middleware.

If the functions are examined in more detail, with the append history function, the heading, position and classification information of the tracked objects are passed by the distance and speed filters, and the history is appended in the related Python dictionaries. With the update history function, the untracked object is deleted from the dictionary and only vehicle and pedestrian type entries are entered into the model.

Heading data entered as input to the model is obtained from both the direction information coming from the segmentation output and the direction information found from the ego vehicle position. Although the segmentation output gives the right direction of the vehicle, due to errors emerging from lidar segmentation, it often reverses the direction by 180 degrees. While heading from the position information, the heading value is incorrect, especially when the vehicle is maneuvering. If the heading value from the position information and the value from the segmentation is more than 90 degrees, the direction information from the segmentation is rotated 180 degrees, thus a more robust heading information structure is created. In addition, a velocity and attention radius filter has been created for the tracked objects, and objects that are far from the ego vehicle or at very low

speeds from the detection system do not enter the model so that the algorithm works more efficiently. The predicted trajectory information from the model was converted into a message type suitable for the planning module and visualized. A configuration file was created in order to easily understand and configure message types and Rostopics which is the name of Ros messages.

Apart from Rostopics and message types, the behavioral prediction module can be configured with various parameters to operate efficiently under different conditions. These parameters include:

- History time step: This parameter determines the number of past time steps to be retained. The perception system operates at 10 frames per second, and since a 2-second history is commonly used in the literature, it has been set as 20 time steps.

- Attention radius: It represents the distance of objects in the vicinity of the ego vehicle in meters. Due to the perception system's error rate being low in the vicinity of the vehicle, the vehicle's low speed (30 km/h), and it being deemed sufficient in conducted tests, an attention radius of 30 meters has been chosen.

- Speed Threshold: This is the minimum speed required for environmental objects entering the model, measured in meters per second (m/s). A speed threshold of 1 m/s has been set to prevent very slow objects or erroneous speed data from entering the model.

- Output Tag: This parameter specifies which model to use. Options include Base, Dynamic Integration, Dynamic Integration + Maps, Dynamic Integration + Maps + Robots Future. Due to the absence of vehicle dynamics and map information in the collected dataset, the base model has been used.

- Predict Horizon: It defines the time length for prediction in seconds. Since it is commonly used in the literature and provides an adequate amount of time for planning, a 6-second duration has been chosen.

These parameters allow for the customization and optimization of the Behavioral Prediction module to suit different scenarios and operational requirements.

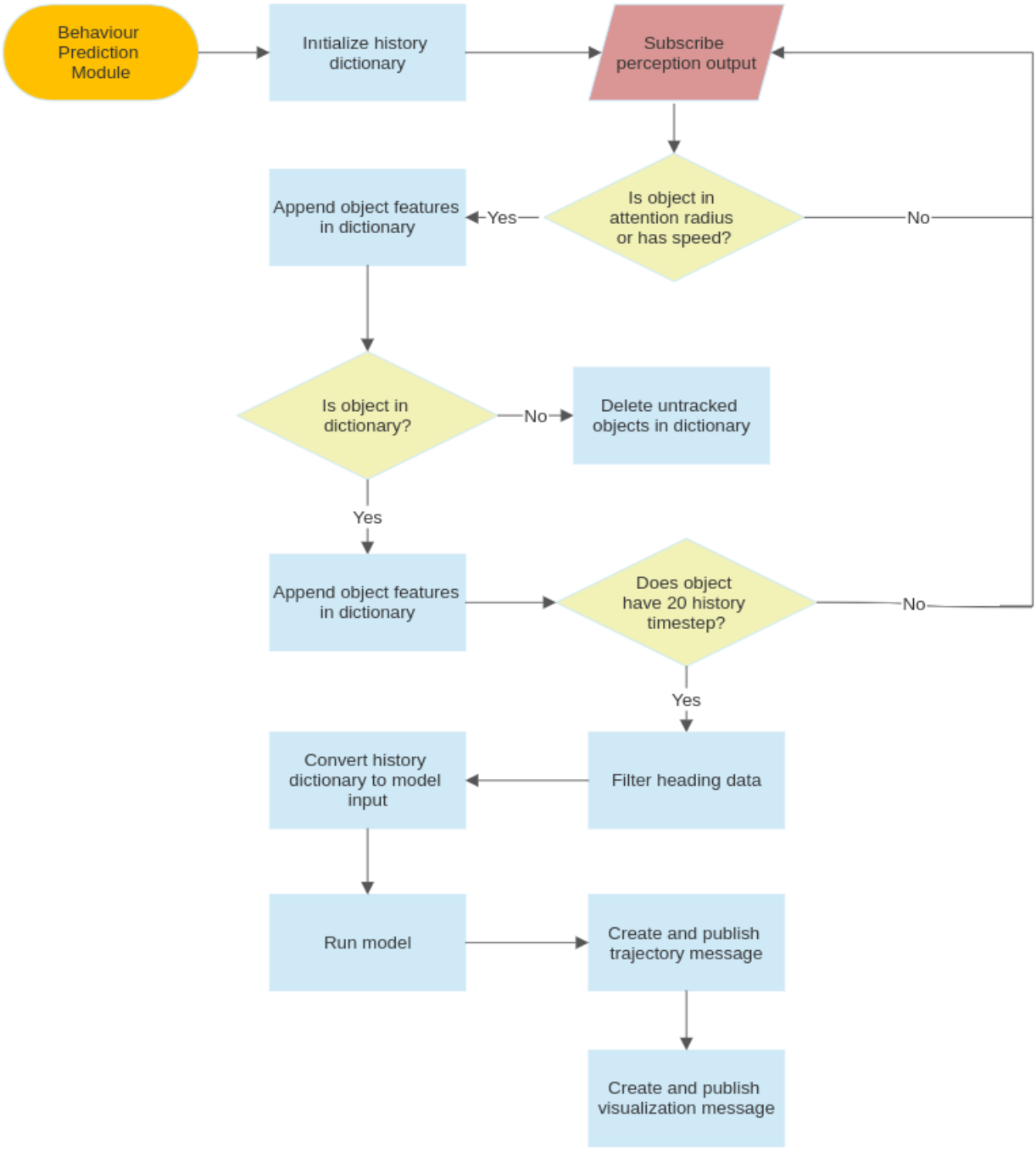

Figure 3.2 Flowchart of Behavioral prediction module

- Pseudo code of behavioral prediction module;

```
subscribe perception output
initialize history dictionary
while ROS is UP
      if object is in attention radius and has speed
            Append object id, x, y, heading, yaw in dictionary
      if object is in dictionary
            Append object id, x, y, heading, yaw in dictionary
      else
            delete object dictionary
      if object has 20 history timestep
            filter heading data
            convert history dictionary to model input
            input object history to model
      create trajectory message from model output
      create visualization marker from model output
      publish trajectory message
      publish visualization message
```

4. RESULTS

4.1 Performance Evaluation Results

The trajectory prediction module was tested by replaying the Rosbag dataset on a laptop equipped with a T1000 graphics card, an i7 9th Gen. Processor, and 16 GB of RAM. The developed software was executed in a Docker and Conda virtual environment with CUDA 11.3 and PyTorch.

The collected Rosbag data was analyzed separately for highway conditions (Sabancı Boulevard) and inner-city conditions (Mustafa Kemal District). The urban segment covers the time interval from the first to the eighth minute of the Rosbag, while the highway segment spans from the eighth to the thirteenth minute. Trajectory predictions were generated and visualized using green sphere markers by executing the trajectory prediction ROS module on the Rosbag data.

As depicted in Figures 4.1, 4.2, and 4.3, during urban driving, the vehicle frequently encountered densely populated environments. In these figures, images captured by the vehicle's top-mounted camera are displayed on the right side, while the outputs of the behavioral prediction module have been juxtaposed with the perception and localization module outputs of the vehicle on the left side.

When the figures from dense environments were inspected, although the speed and attention radius filter was added, incorrect speed information from the perception layer for the standing vehicles caused the algorithm to work slowly in a dense environment. Sudden changes in the direction of vehicles and pedestrians in dense areas are a factor that reduces the performance of the behavior prediction module.

In Figure 4.1, the route prediction of the vehicles around the autonomous vehicle was made when approaching the intersection in the Mustafa Kemal district. As can be seen in the world model on the left side, due to the filter used, while there is no prediction for stopping vehicles, trajectory prediction estimates of vehicles turning at the intersection have been made.

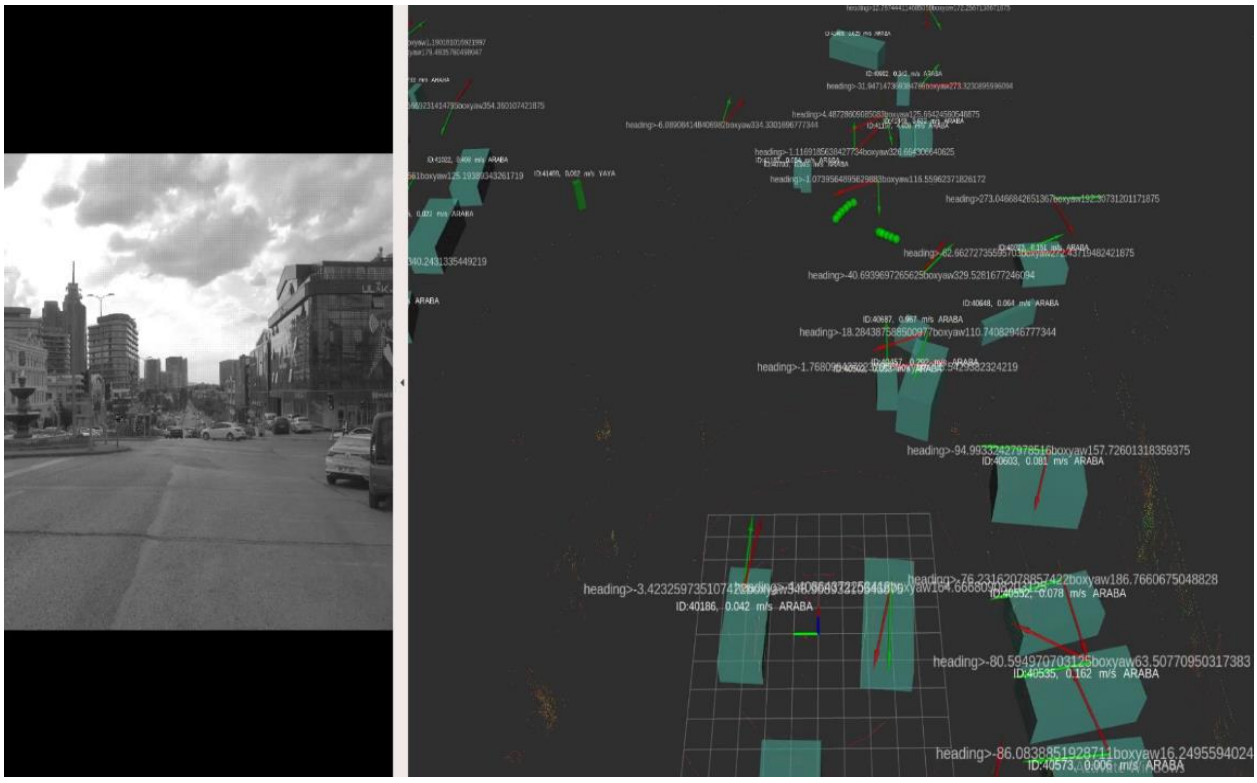

Figure 4.1 Visualization of dense environment trajectory prediction-1

In Figure 4.2, route estimates of pedestrians crossing the street were made. Pedestrians are also difficult to track because they are small in size compared to vehicles. For this reason, the possible route of only one of the pedestrians crossing the street could be estimated.

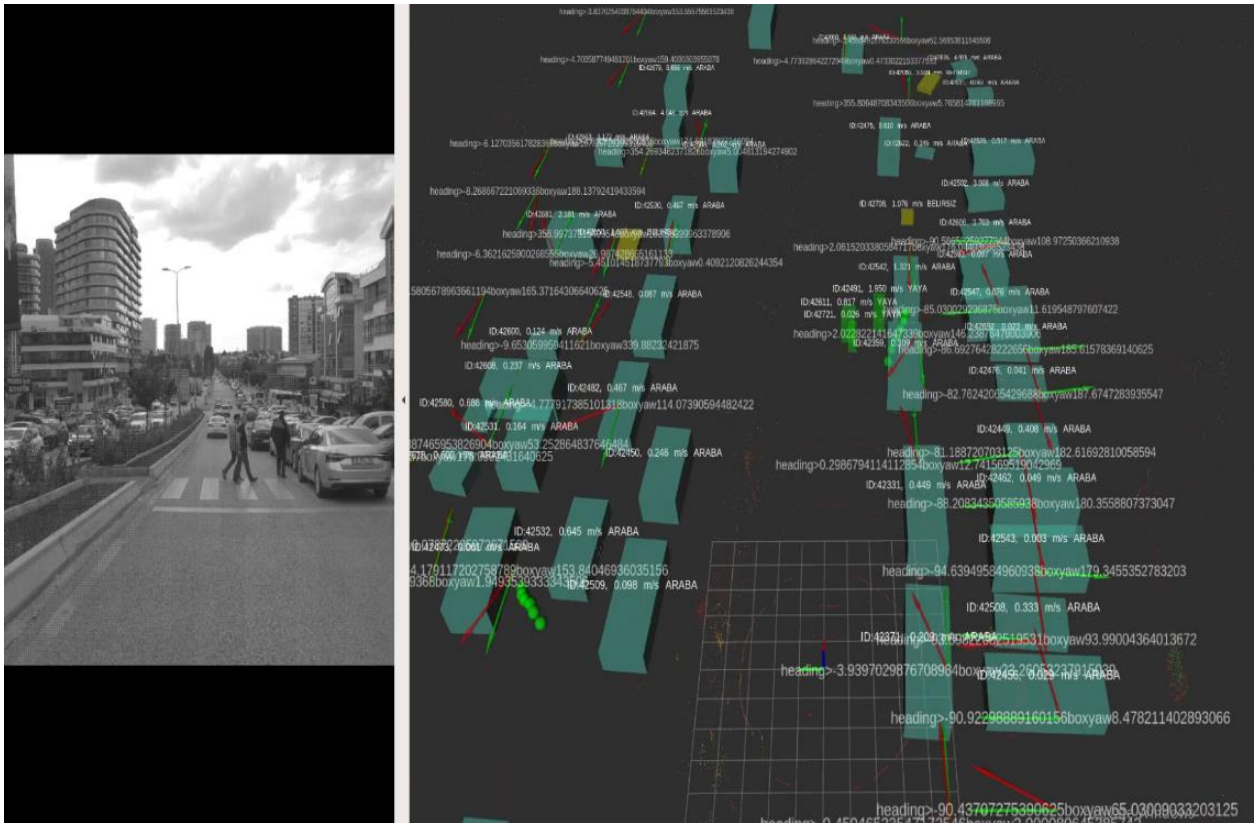

Figure 4.2 Visualization of dense environment trajectory prediction-2

In Figure 4.3, route estimates of the vehicle on the left side and the vehicle coming from the opposite lane were made. As the vehicle approaches the intersection, the route prediction of the turning vehicles in the opposite lane, the vehicles in the back cross and the vehicle in front are made in Figure 4.4.

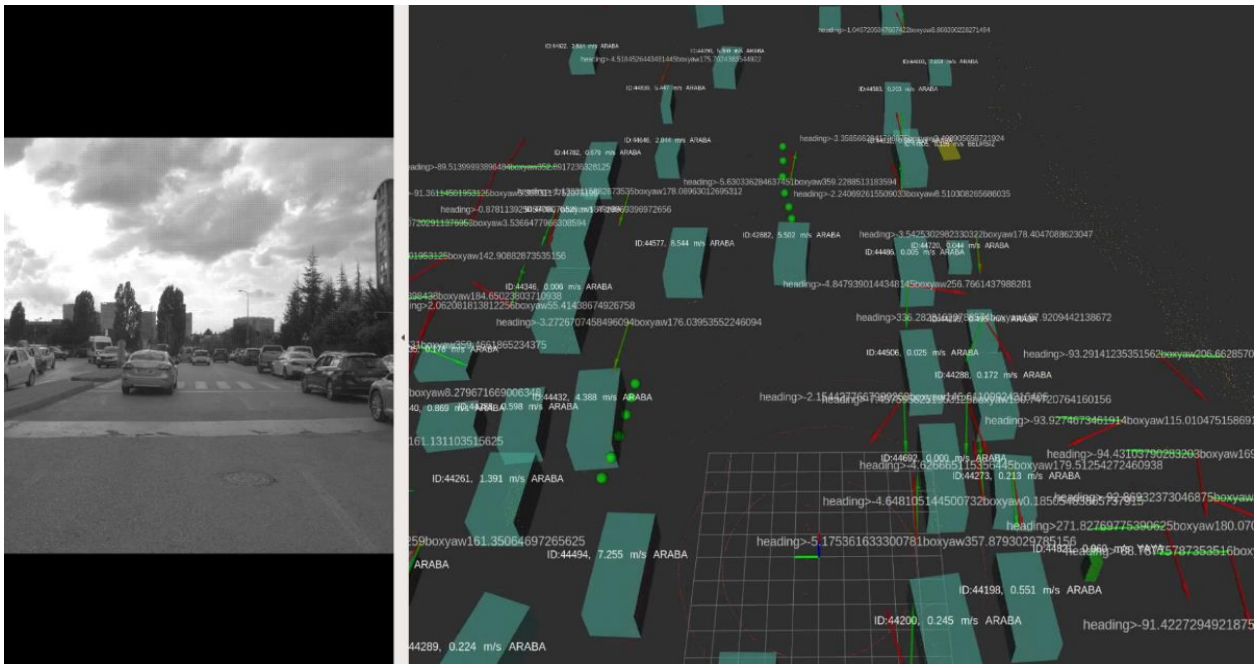

Figure 4.3 Visualization of dense environment trajectory prediction-3

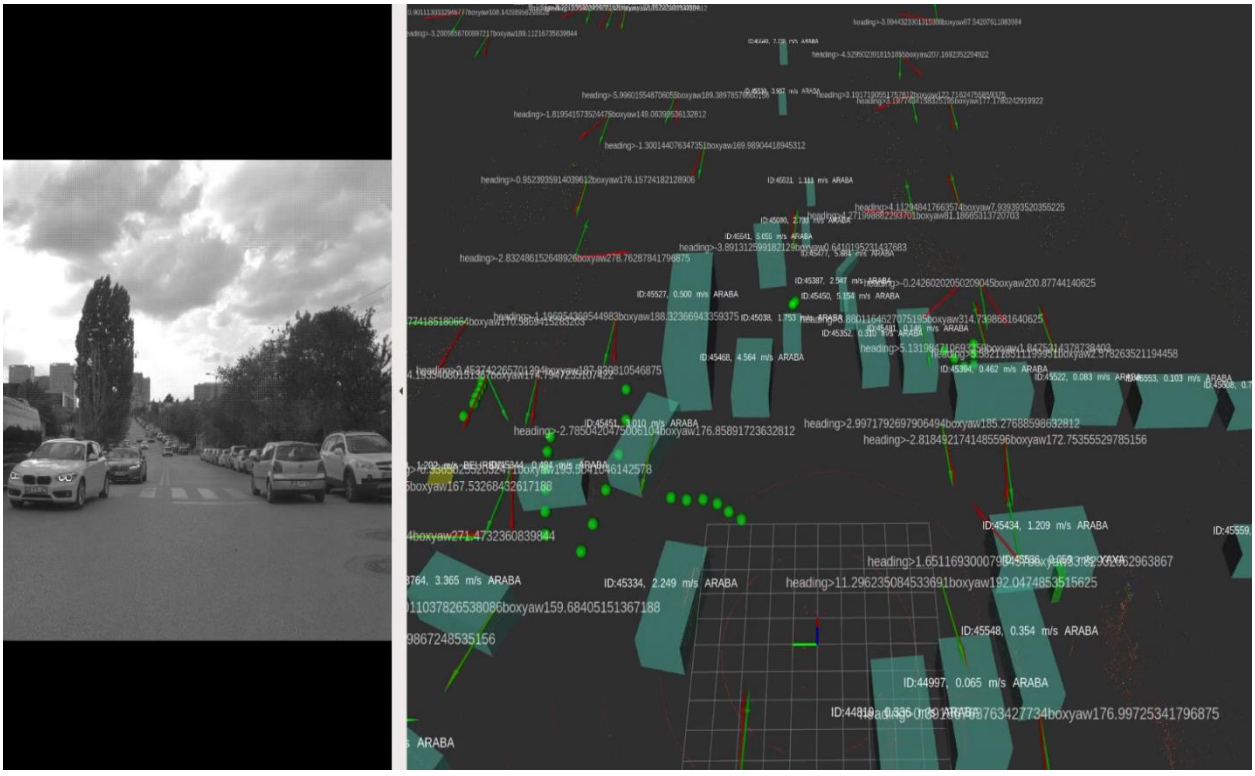

Figure 4.4 Visualization of dense environment trajectory prediction-4

In Figures 4.5 and 4.6, while the vehicle was being driven at a speed of about 50 km/h on Sabancı Boulevard, the route estimation of multiple vehicles displayed on the left side was made. The estimated routes are quite close to the actual routes of the vehicles.

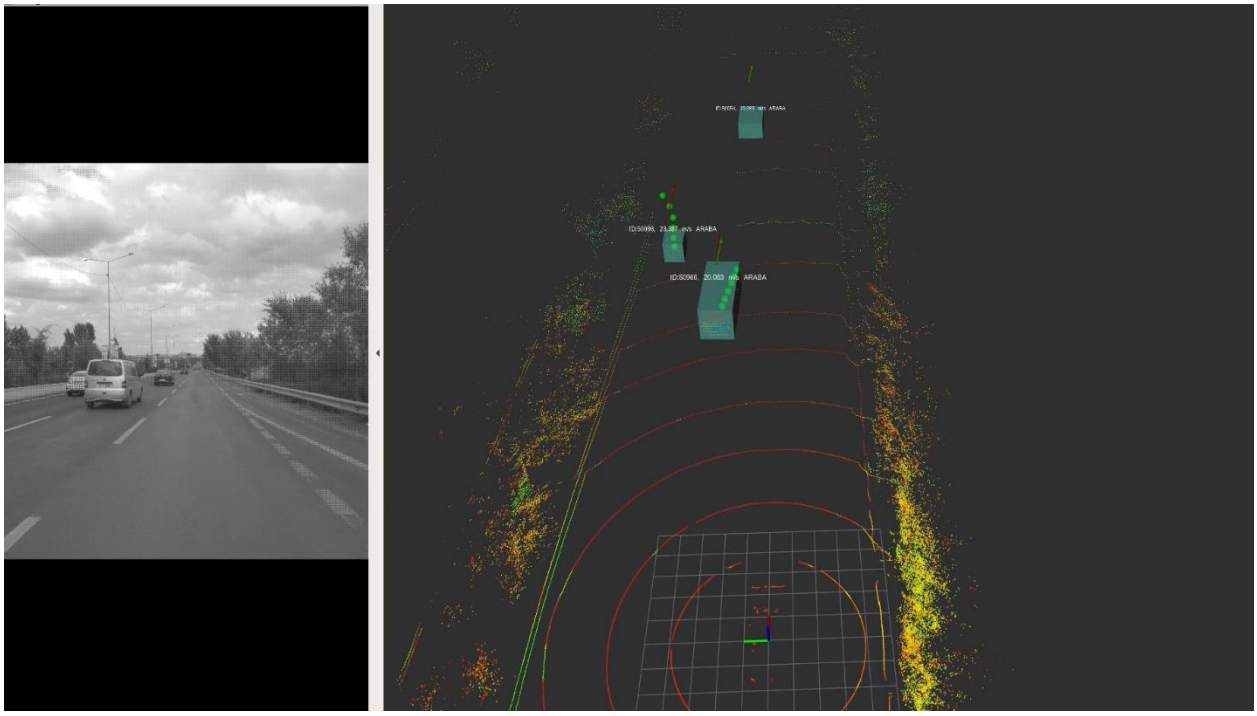

Figure 4.5 Visualization of multi vehicle trajectory prediction-1

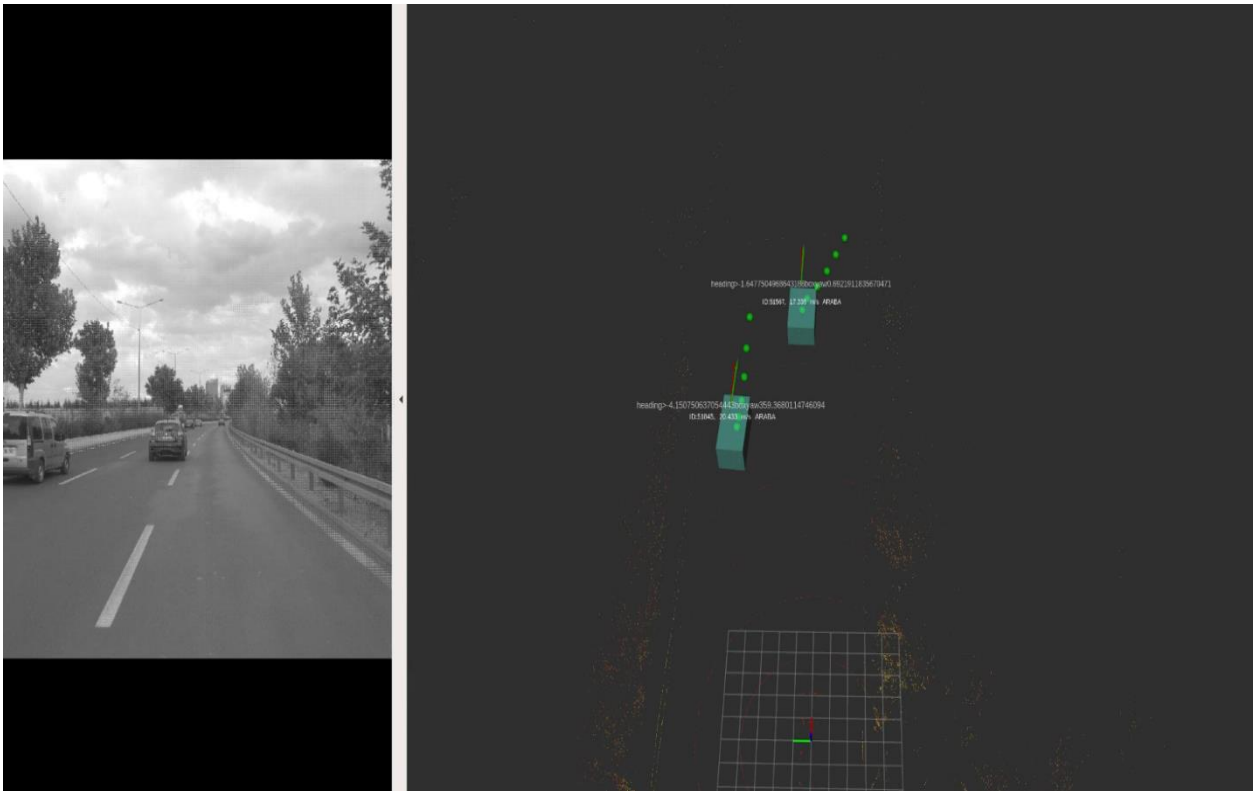

Figure 4.6 Visualization of multi vehicle trajectory prediction-2

Rosbag's highway condition driving between 8 min and 13 min has achieved better results compared to the urban section. Approximately 5 km of driving has been done under highway conditions. Nine vehicles have been tracked by perception without error. Behavioral estimation of nine vehicles in highway condition from Figure 4.7 to 4.15, respectively, was performed. The green dots on the figures indicate the possible routes of the vehicles. In addition, as seen in Figures 4.5 and 4.6, the trajectory predictions of two or three vehicles have been made at the same time without loss of performance.

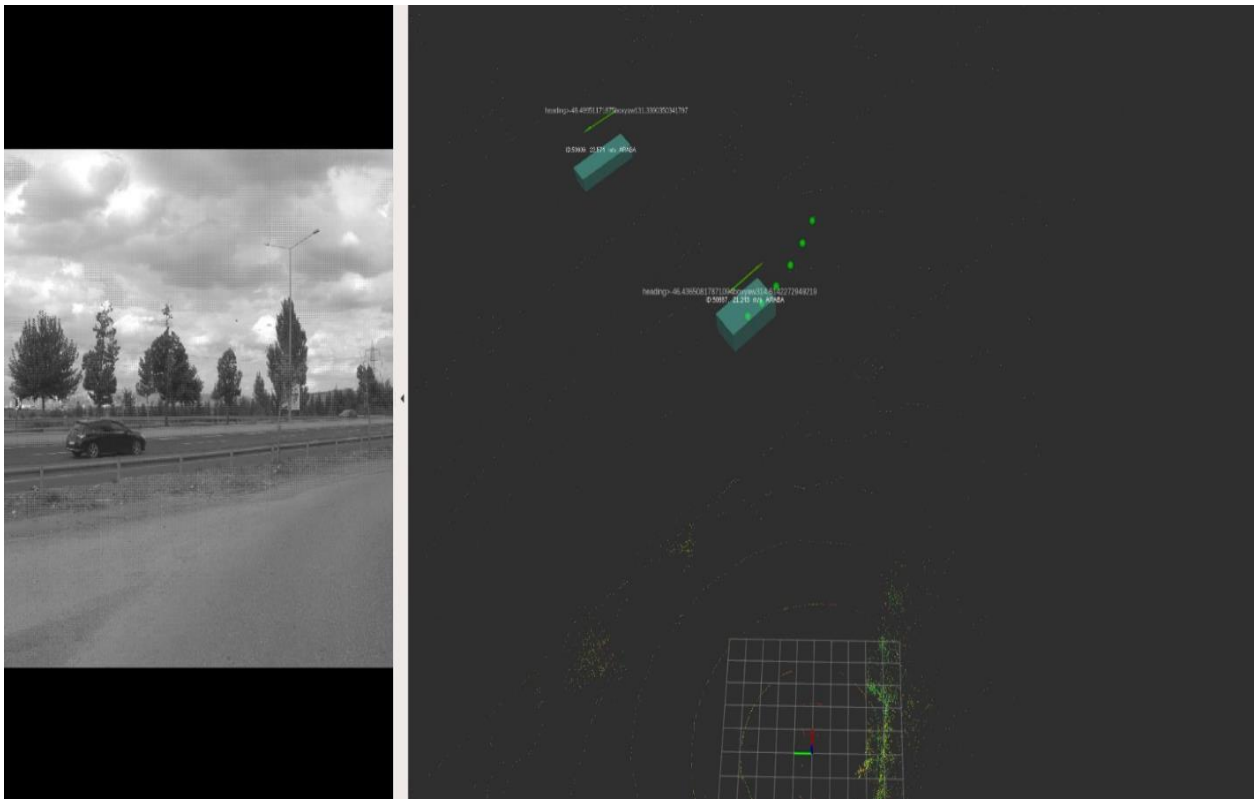

Figure 4.7 Visualization of vehicle-1 trajectory prediction

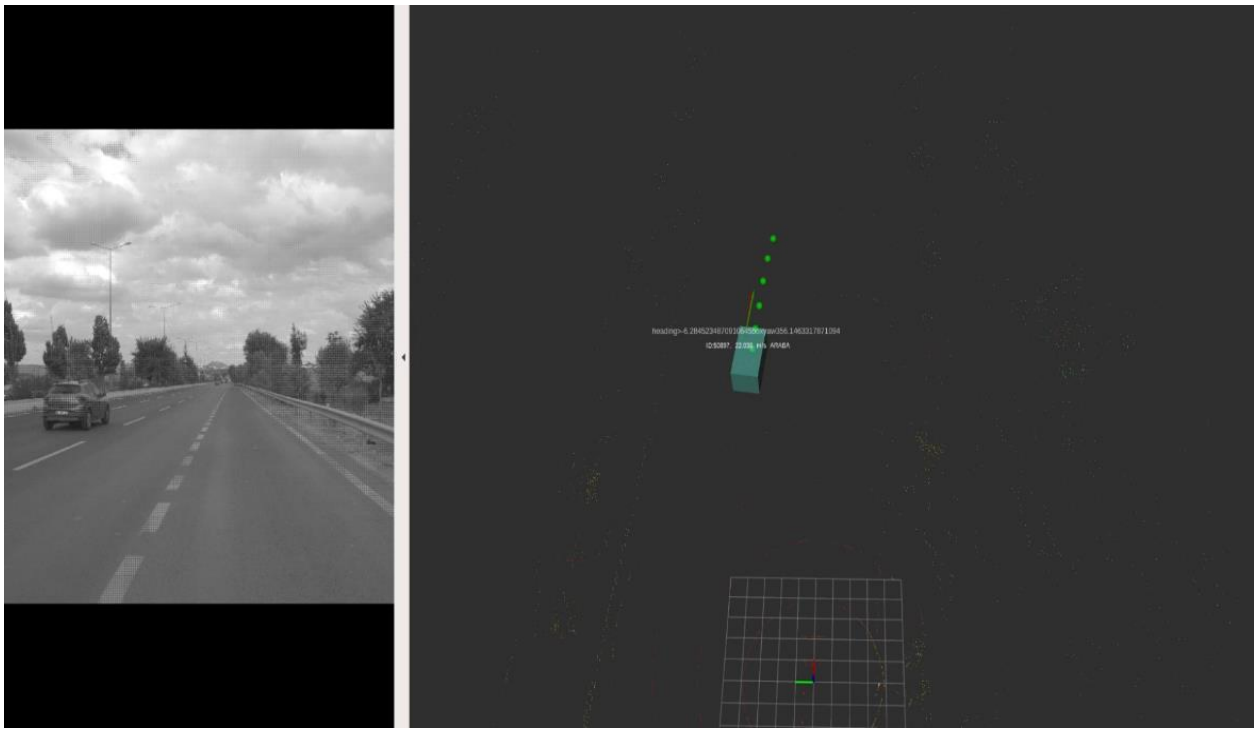

Figure 4.8 Visualization of vehicle-2 trajectory prediction

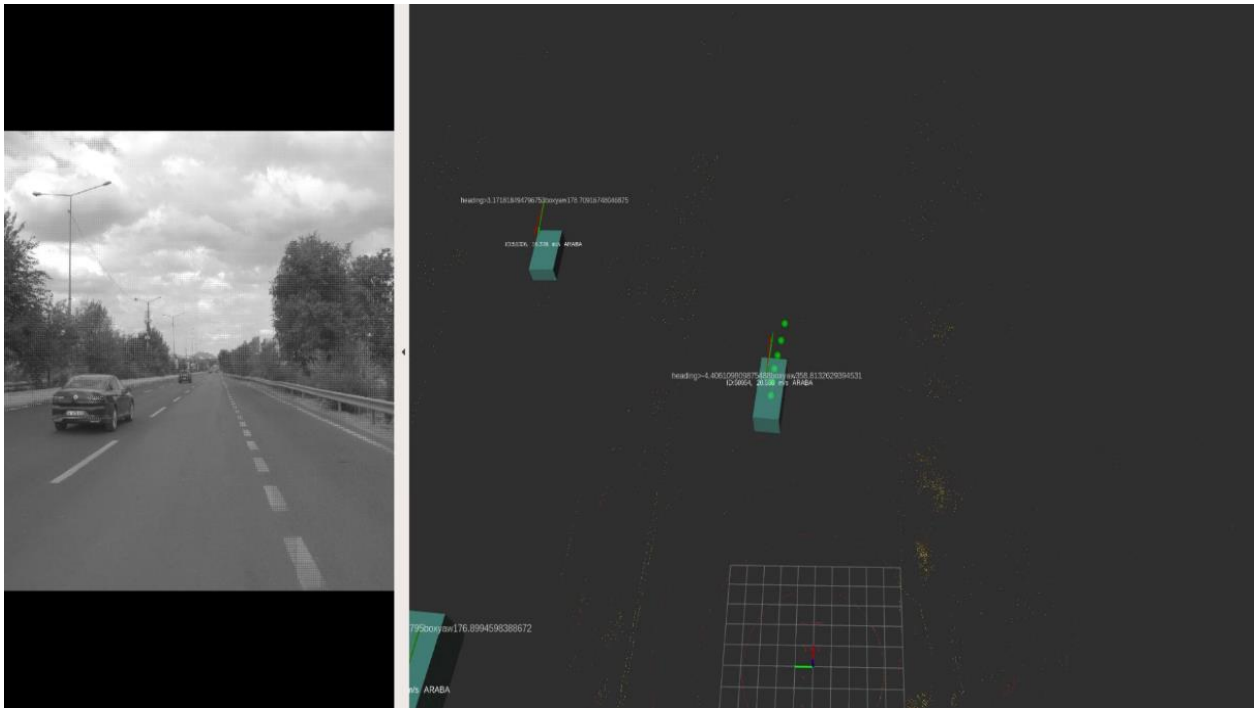

Figure 4.9 Visualization of vehicle-3 trajectory prediction

In Figure 4.7, the autonomous vehicle was tracked while leaving the main road to Sabancı Boulevard and route prediction was made. In Figure 4.8 and Figure 4.9, the route prediction of vehicles leaving the left lane has been successfully performed. In Figure 4.10 and Figure 4.11, the behavior prediction output of larger commercial vehicles compared to passenger cars was visualized. In Figure 4.15, the behavior prediction output of a pickup truck is visualized.

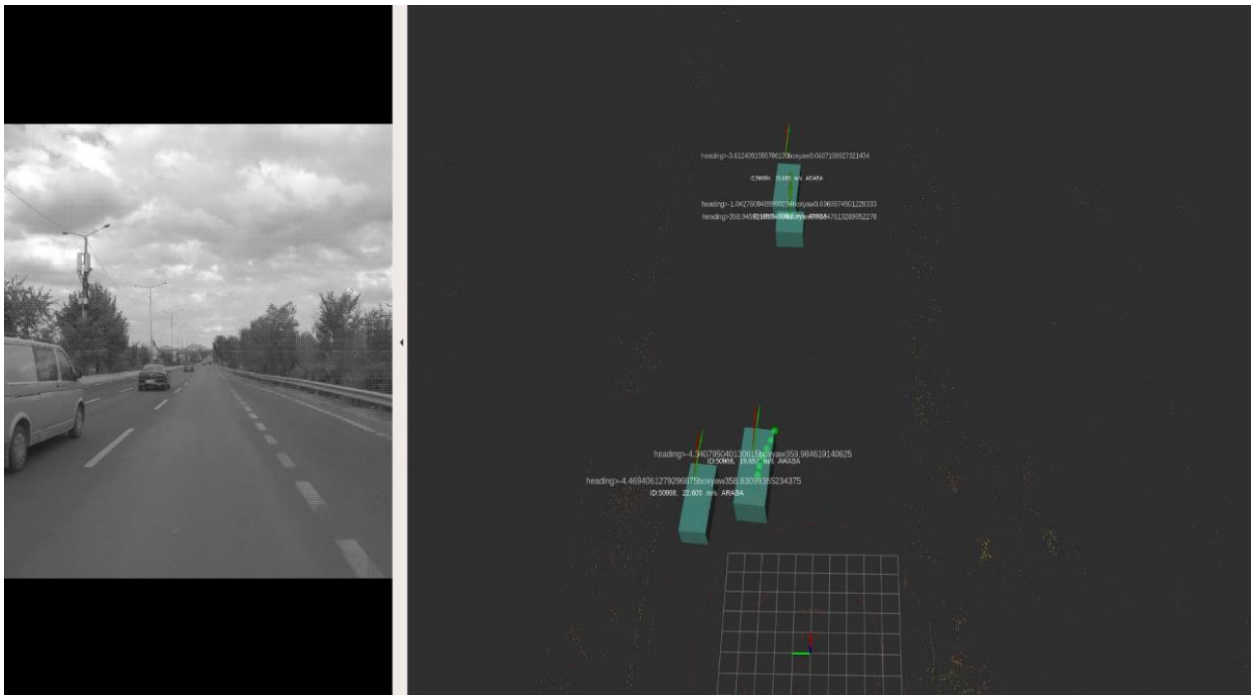

Figure 4.10 Visualization of vehicle-4 trajectory prediction

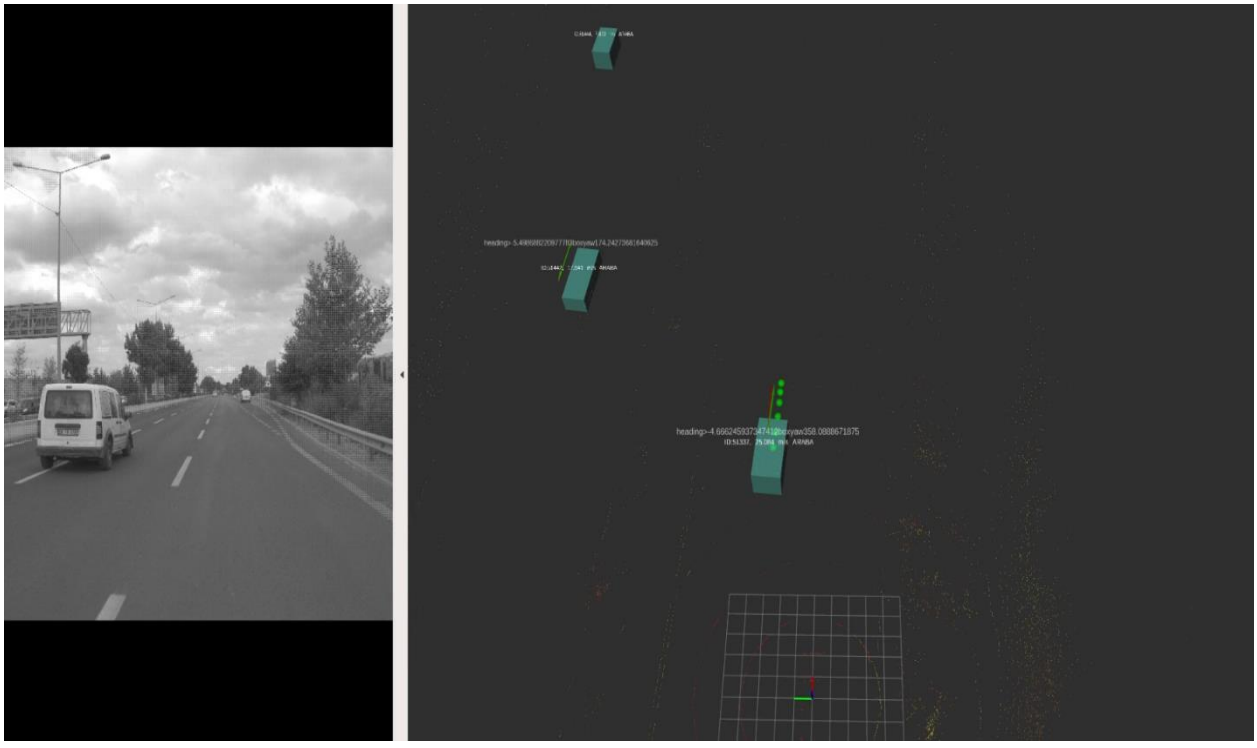

Figure 4.11 Visualization of vehicle-5 trajectory prediction

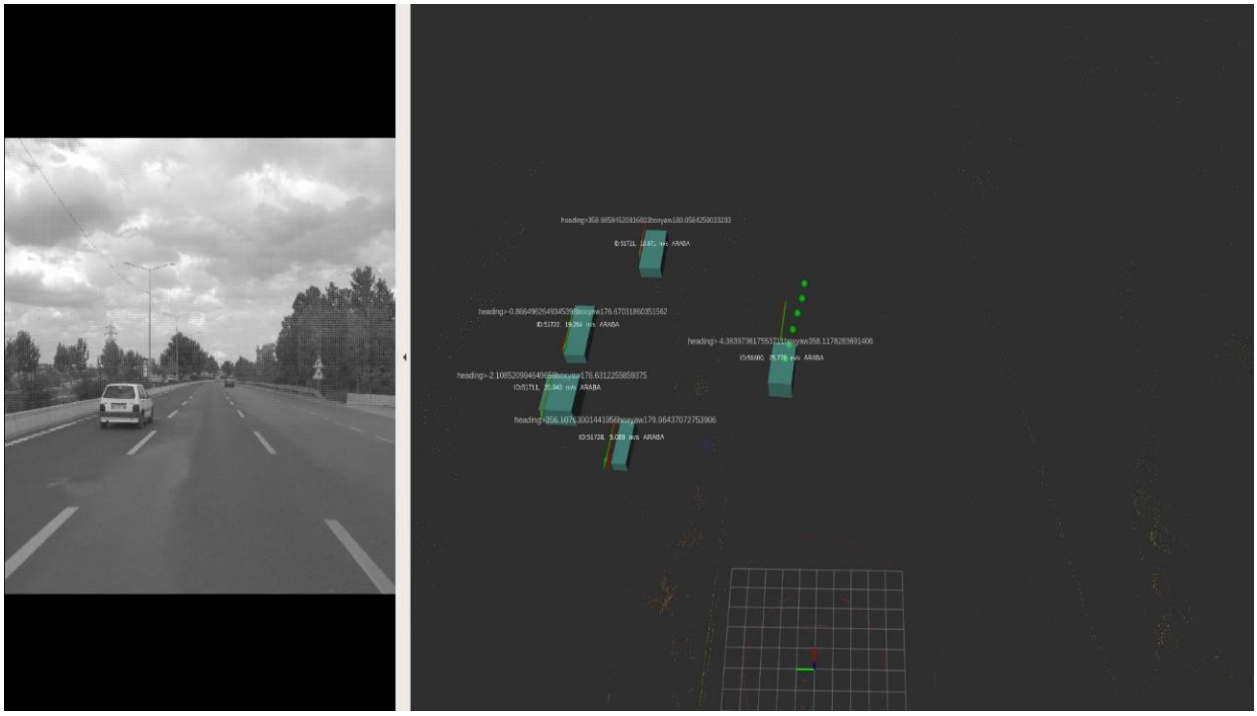

Figure 4.12 Visualization of vehicle-6 trajectory prediction

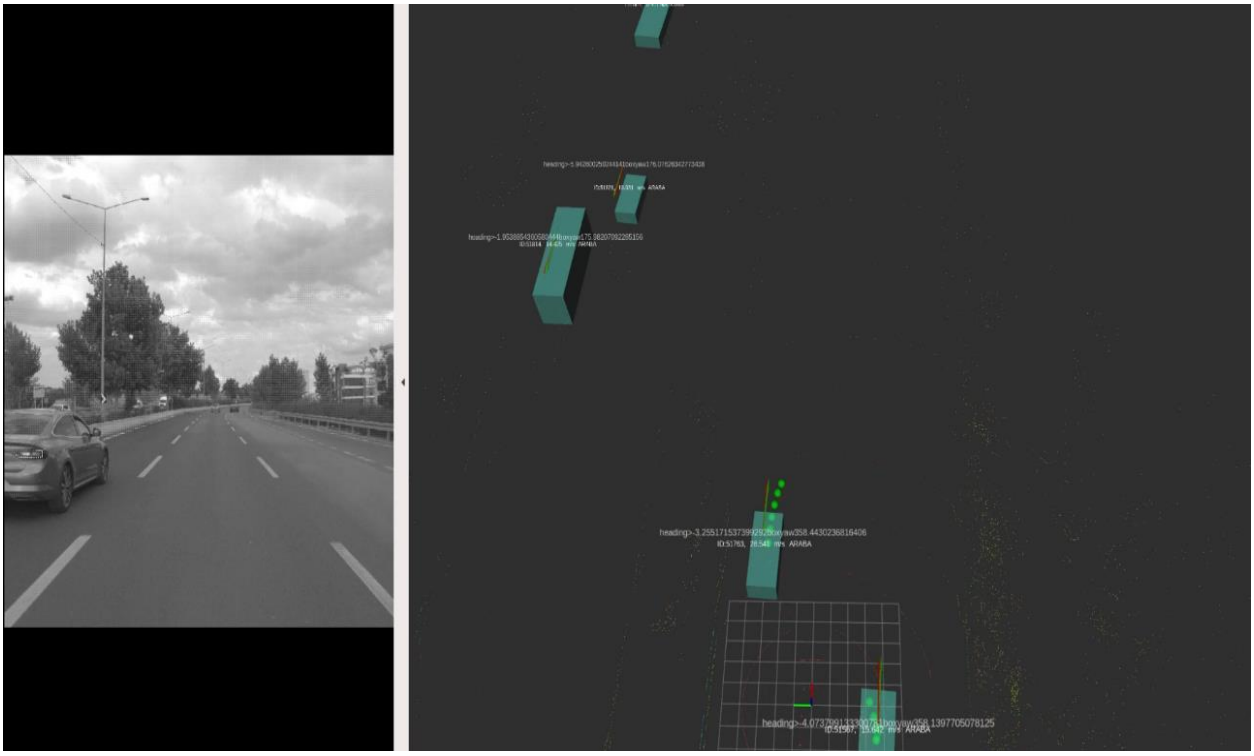

Figure 4.13 Visualization of vehicle-7 trajectory prediction

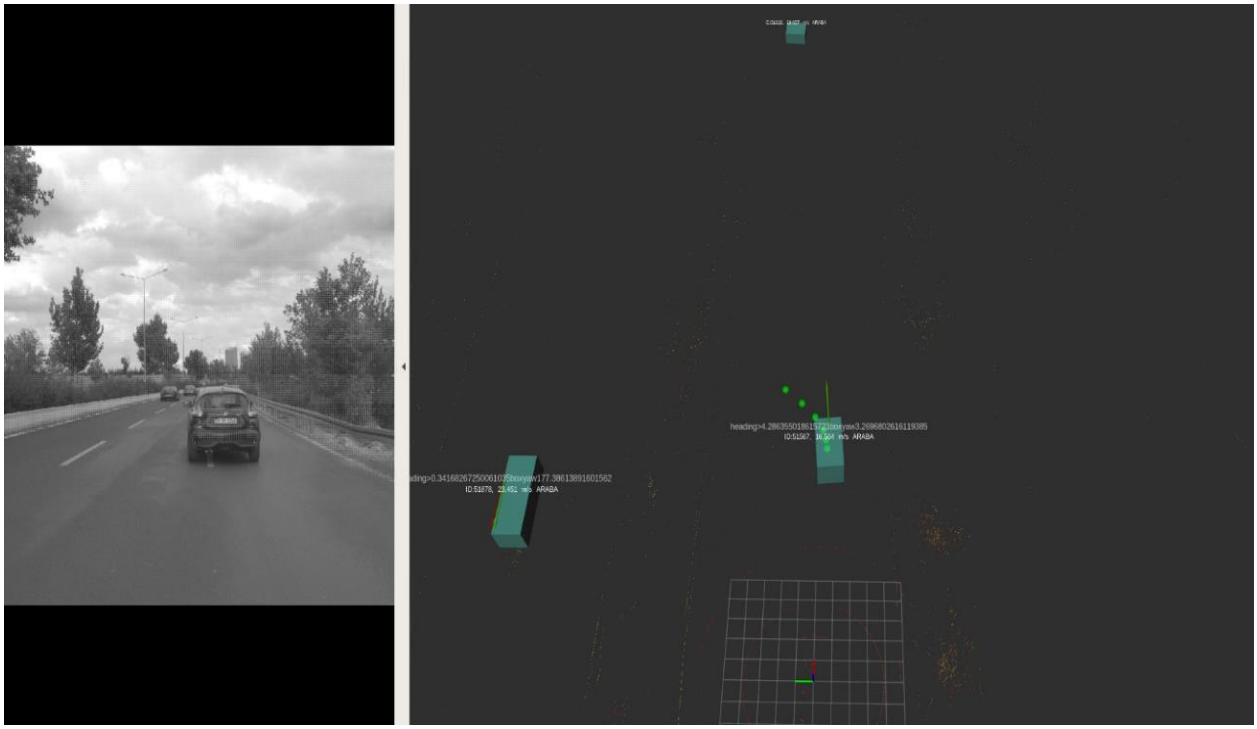

Figure 4.14 Visualization of vehicle-8 trajectory prediction

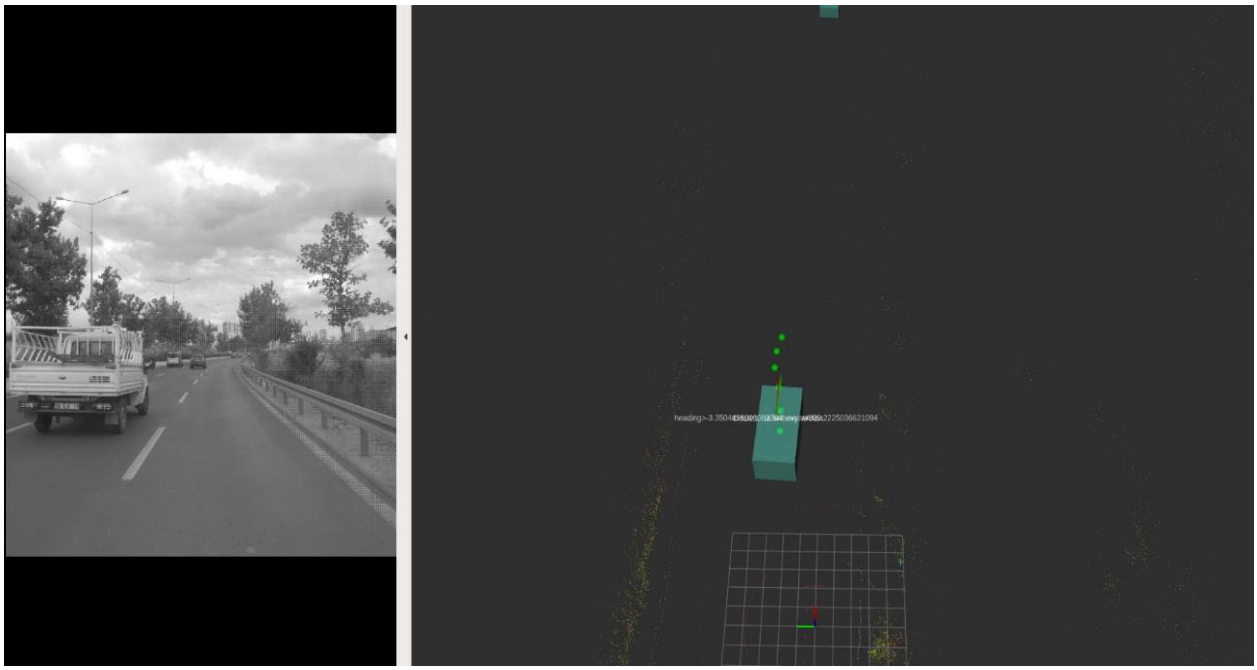

Figure 4.15 Visualization of vehicle-9 trajectory prediction

The outcomes of the Trajectron++ model are presented in Table 4.1. The model underwent testing on the test dataset of the nuScenes dataset on which it was trained. As outlined in the table, according to these results, the FDE for vehicles is 0.42 meters in the first second, and it increases to 2.48 meters by the fourth second.

In the Trajectron++ paper [4], the authors assert, based on their experience, that the error of the detection system in the nuScenes dataset ranges from 22 to 24 cm and subtracts this error margin from their own results. In Table 4.1, since the Rosbag dataset results encompass the detection system error, 24 cm has been incorporated into the nuScenes dataset results. Upon comparing these results with those derived from Rosbag testing, it becomes apparent that the disparity, approximately 0.5 meters at the first second, intensifies with each subsequent second, ultimately reaching 2.5 meters by the fourth second.

Table 4.1 FDE(2) results of Trajectron++ model on nuScenes and Rosbag Dataset

Methods	***FDE(m)***							
	Vehicle-only							
	nuScenes				***Rosbag Dataset (Our Dataset)***			
	1s	***2s***	***3s***	***4s***	***1s***	***2s***	***3s***	***4s***
Trajectron++	0.42	0.81	1.49	2.48	0.92	1.87	3.70	4.85
Trajectron++ +∫, M	0.07	0.69	1.38	2.44	-	-	-	-

Legend: ∫ = Considering Dynamics , M = Map Encoding

The average RMSE of these nine vehicles and the RMSE of five vehicles of 5 seconds are given in Table 4.2. The longer the estimation period, the greater the amount of error. Similar to the urban roads section, although the incorrect perception output or the inability of the perception module to track the surrounding vehicle is a factor that reduces my performance, erroneous data has not been received from the perception module very often. One of the reasons for incorrect data coming from the perception module is incorrect data coming from GPS and IMU. However, since the data was collected at an off-peak time of the day, there was a few erroneous data.

Table 4.2 RMSE (1) results of tested Rosbag

Objects	***RMSE(m)***				
	1s	***2s***	***3s***	***4s***	***5s***
Vehicle1	0.87	2.13	4.14	5.68	7.25
Vehicle2	0.92	2.09	4.28	5.82	6.43
Vehicle3	0.70	1.90	3.91	5.11	6.12
Vehicle4	1.20	2.42	4.72	6.23	8.04
Vehicle5	0.78	1.75	3.78	5.02	6.10
Vehicle6	1.45	2.10	3.99	4.97	7.05
Vehicle7	0.75	2.78	3.41	6.01	7.73
Vehicle8	1.28	2.35	4.66	5.28	6.87
Vehicle9	1.12	1.99	4.03	6.36	8.80
Overall	1.07	2.17	4.10	5.61	7.15

Table 4.3 displays the FDE, ADE, and MR outcomes for the nine vehicles that were accurately tracked by the detection module over a 5-second interval. It was determined that the average FDE result is approximately 7 meters, the average ADE result is 4 meters, and the average MR result is 70 percent.

Table 4.3 FDE (2), ADE (3) and MR (4) results of tested Rosbag

Objects	***FDE(m)***	***ADE(m)***	***MR(m)***
Vehicle1	8.52	5.78	0.75
Vehicle2	9.12	6.89	0.85
Vehicle3	7.88	4.45	0.77
Vehicle4	6.56	3.24	0.62
Vehicle5	5.14	3.15	0.62
Vehicle6	5.82	2.01	0.76
Vehicle7	6.11	4.46	0.64
Vehicle8	7.15	5.14	0.70
Vehicle9	4.77	2.03	0.68
overall	6.78	4.12	0.71

4.2 Findings

Throughout the tests, when a single vehicle or pedestrian is present within the environment, the module operates at 6 fps. As environmental factors increase, the module's operating frequency inversely decreases relative to the number of factors. Specifically, the system operates at 6 fps with one environmental object and 3 fps with four environmental objects. Under the condition that the vehicle is moving at a maximum speed of 30 km/h, and considering that other autonomy modules such as planning and perception can operate smoothly at 5 fps, as well as taking into account the more powerful processor of the onboard computer, it is anticipated that the system will function without speed issues in environments containing up to four objects. In order not to reduce the performance, model parallelization methods such as converting TensorRT can be used. However, the TensorRT library does not yet support the conversion of some modules of the Trajectron++ model, such as GRU and GMM.

The inclusion of an attention radius and a speed filter prevented distant, stationary, or very slow objects from entering the model, leading to improved model efficiency.

The discrepancy between the results presented in Table 4.1 of the Trajectron++ paper and the outcomes from Rosbag testing arises from multiple factors. These factors encompass the nuScenes dataset being collected in a more organized traffic setting compared to the collected

Rosbag data, the higher error rate of the detection system within the Rosbag data, and the greater velocities of tracked vehicles in the Rosbag data.

Considering that the distance between the two lines is 3 m for side roads and 3.50 m for main roads, and taking into account that lane changes typically occur over 2-4 seconds, the results for highway vehicles provided in Table 4.2 suggest that vehicles up to two seconds exhibit an acceptable level of RMSE.

RMSE can be reduced by getting more accurate data from the perception layer. Speed and heading information, which is one of the data from the perception layer, comes to the perception layer from the localization layer and it comes to the localization layer from GPS and IMU sensors. Therefore, accurate data should be received from GPS and IMU sensors for the successful performance of the behavior prediction module; otherwise, incorrect results will emerge from the speed and heading data entered as input to the model. However, the results in Table 4.3 also support the results in Table 4.2. According to Table 4.3 results, it has been observed that the algorithm is prone to error in real life during long prediction times.

During drives in urban conditions, as depicted in the dense environment from Figure 4.1 to 4.4, inputs are simultaneously fed into the model, thereby reducing the model's operational speed. However, the fact that the route that both pedestrians and vehicles will take is more uncertain than highway roads is one of the negative factors affecting this structure in urban use. Similar to highway roads, speed and heading information from GPS and IMU are also important in urban areas. Even in the city, since the speeds of vehicles are lower, errors in speeds affect the system more. Although the slow speed makes it difficult to find the heading data correctly, more accurate heading data can be obtained thanks to the added binary heading data structure. In order for the behavior prediction module to be used in the city, the environment must be controlled and run on more advanced computers. Controlled environment refers to the environments in which the algorithm is fed with HDMap, traffic rules, traffic signs, node interactions and objects move within the framework of these rules.

In the field of behavioral prediction, as it can be understood from the related work section, many models have been studied and continue to be studied. Although the studied models generally give successful results for the test sets of their own datasets, their speed and performance decrease in dense environments such as urban areas. Successful results can be obtained by creating more complex datasets and developing better model architectures. However, especially since pedestrians can change direction suddenly, people even in real life have difficulty predicting the possible

movement of pedestrians and vehicles. Therefore, the behavior prediction problem is a difficult problem for robots as well as for living beings. Especially as the estimated time gets longer, the results are moving away from availability. In order for the behavior prediction module to be used in public, the environment must be controlled, that is, the map must be used, and the surrounding vehicles and pedestrians must act in accordance with the traffic rules and the map.

From a broader perspective apart from transportation vehicles, this software can be used in any autonomous ground vehicle such as health care robots, agricultural robots, shuttle services etc. and it increases the accuracy of planning module. In the future, it is planned to add ConvLSTM to the model to increase accuracy and accelerate this model using TensorRT.

5. CONCLUSION

Since the possible routes of environmental objects will be taken into consideration, with the behavioral prediction module obtained as a result of this study, the ratio of an autonomous vehicle making a mistake in the environment of moving objects can be significantly reduced. An end-to-end, real-time and robust behavioral prediction structure has been established from the detection module to the planning module, with contributions such as writing real-time inference to the current model, passing it to the ROS infrastructure and correcting or filtering faulty data.

Once for all, when the literature is examined, although there are autonomy modules such as open-source localization planning perception, there is no multi-model real-time behavior prediction module working with ROS. However, Trajectron++ is used as the model; also, there is no open-source study for end-to-end integration of a model into an autonomous vehicle. The study is innovative in these aspects. The module works more efficiently and robustly due to the added history hold/drop structure and data filtering and correction features. In order to test the developed module, data were collected by traveling both in the city and on the highway conditions with an autonomous vehicle. The developed module has been tested on the data collected by an autonomous vehicle, and RMSE, FDE, ADE and MR results are calculated and shown.

In this study, trajectory prediction methods were examined, compared with each other, and as a result of these comparisons, graph-structured structures were seen to be more successful. Afterwards, the data was collected and the content of the data was explained. The developed software is explained, tested and the results are expressed.

REFERENCES

[1] Koubaa, A., "Robot Operating System (ROS): The Complete Reference (Volume 2)", *Springer*, 2017, doi: 10.1007/978-3-319-54927-9.

[2] Hintjens, P., "ZeroMQ: Messaging for Many Applications", *O'REILLY*, CA, 2013.

[3] Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W., "Robot operating system 2: Design, architecture, and uses in the wild", *Sci. Robot.*, 7 (66) (2022), doi: 10.48550/arXiv.2211.07752.

[4] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conference on Computer Vision. Springer*, 2020, pp. 683 700, doi: 10.1007/978-3-030-58523-5_40.

[5] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, 2022.

[6] M. Gulzar, Y. Muhammad, and N. Muhammad, "A survey on motion prediction of pedestrians and vehicles for autonomous driving," *IEEE Access*, 2021.

[7] S. Lefèvre, D. Vasquez, and C. Laugier, ''A survey on motion prediction and risk assessment for intelligent vehicles,'' *ROBOMECH J.*, vol. 1, no. 1, pp. 1–14, 2014.

[8] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, ''What the constant velocity model can teach us about pedestrian motion prediction,'' *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1696-1703, Apr. 2020.

[9] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," *IEEE 5th International Conference on Intelligent Computer Communication and Processing*, 2009.

[10] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," *11th International Conference on Information Fusion*, 2008.

[11] P. Lytrivis, G. Thomaidis, and A. Amditis, "Cooperative path prediction in vehicular environments," *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference*, 2008.

[12] N. Kaempchen, K. Weiss, M. Schaefer, and K. C. J. Dietmayer, "Imm object tracking for high dynamic driving maneuvers," in *Intelligent Vehicles Symposium*, 2004

[13] B. Jin, J. Bo, S. Tao, H. Liu, and G. Liu, “Switched kalman filterinteracting multiple model algorithm based on optimal autoregressive model for maneuvering target tracking,” *Iet Radar Sonar & Navigation*, vol. 9, no. 2, pp. 199–209, 2015.

[14] K. Okamoto, K. Berntorp, and S. Di Cairano, “Driver intention-based vehicle threat assessment using random forests and particle filtering,” *IFAC-PapersOnLine*, vol. 50, pp. 13 860–13 865, 07 2017.

[15] Y. Wang, Z. Liu, Z. Zuo, Z. Li, L. Wang, and X. Luo, “Trajectory planning and safety assessment of autonomous vehicles based on motion prediction and model predictive control,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8546–8556, 2019.

[16] E. Coelingh, A. Eidehall, and M. Bengtsson, ‘‘Collision warning with full auto brake and pedestrian detection—A practical example of automatic emergency braking,’’ in *Proc. 13th Int. IEEE Conf. Intell. Transp. Syst*., Sep. 2010, pp. 155–160.

[17] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, ‘‘Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models,’’ *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5999–6008, Jul. 2018.

[18] Batz, K. Watson, and J. Beyerer, “Recognition of dangerous situ ations within a cooperative group of vehicles,” in *Intelligent Vehicles Symposium*, 2009.

[19] C. Hermes, C. Wohler, K. Schenk, and F. Kummert, “Long-term vehicle motion prediction,” in *Intelligent Vehicles Symposium*, 2010.

[20] P. Trautman and A. Krause, “Unfreezing the robot: Navigation in dense, interacting crowds,” in 2010 *IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE*, 2010, pp. 797–803.

[21] Y. Guo, V. V. Kalidindi, M. Arief, W. Wang, J. Zhu, H. Peng, and D. Zhao, “Modeling multi-vehicle interaction scenarios using Gaussian random field,” in 2019 *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3974–398.

[22] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, “Learning-based ` approach for online lane change intention prediction,” in *IEEE Intelligent Vehicles Symposium*, 2013.

[23] S. Qiao, D. Shen, X. Wang, N. Han, and W. Zhu, “A self-adaptive parameter selection trajectory prediction approach via hidden markov models,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 284–296, 2015.

[24] Q. Deng and D. Soffker, “Improved driving behaviors prediction ¨ based on fuzzy logic-hidden markov model (fl-hmm),” in 2018 *IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 2003–2008.

[25] T. Gindele, S. Brechtel, and R. Dillmann, ‘‘A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments,’’ in *Proc. 13th Int. IEEE Conf. Intell. Transp. Syst.,* Sep. 2010, pp. 1625–1631.

[26] Lee, N., Choi, W., Vernaza, P., Chor, C. B., Torr, P. H. S., And Chandraker, M. K. “DESIRE: distant future prediction in dynamic scenes with interacting agents.” *CoRR* abs/1704.04394, 2017.

[27] S. Dai, L. Li, and Z. Li, ‘‘Modeling vehicle interactions via modified LSTM models for trajectory prediction,’’ *IEEE Access,* vol. 7, pp. 38287–38296, 2019.

[28] F. Diehl, T. Brunner, M. T. Le, and A. Knoll, “Graph neural networks for modelling traffic participant interaction,” in 2019 *IEEE Intelligent Vehicles Symposium (IV),* 2019.

[29] X. Li, X. Ying, and M. C. Chuah, “Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving,” *arXiv preprint arXiv*:1907.07792, 2019.

[30] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, “The apolloscape open dataset for autonomous driving and its application,” in 2018 *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* 2018.

[31] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social gan: Socially acceptable trajectories with generative adversarial networks,” in 2018 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* 2018.

[32] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” *Advances in neural information processing systems,* vol. 28, pp. 3483–3491, 2015.

[33] L. Sun, W. Zhan, and M. Tomizuka, “Probabilistic prediction of interac tive driving behavior via hierarchical inverse reinforcement learning,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE,* 2018, pp. 2111–2117.

[34] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, “Imitating driver behavior with generative adversarial networks,” in *2017 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2017,* pp. 204–211.

[35] S. Choi, J. Kim, and H. Yeo, “Trajgail: Generating urban vehicle trajec tories using generative adversarial imitation learning,” *Transportation Research Part C: Emerging Technologies,* vol. 128, p. 103091, 2021.

[36] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum entropy deep in verse reinforcement learning,” *arXiv preprint arXiv*:1507.04888, 2015.

[37] C. You, J. Lu, D. Filev, and P. Tsiotras, “Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning,” *Robotics and*

Autonomous Systems, vol. 114, pp. 1–18, 2019.

[38] C. Jung and D. H. Shim, "Incorporating multi-context into the traversability map for urban autonomous driving using deep inverse reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1662–1669, 2021.

[39] https://github.com/enyen/Deep-Trajectory-Prediction (Accessed: 20 April 2023)

[40] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW), 2018.

[41] V. Lefkopoulos, M. Menner, A. Domahidi, and M. N. Zeilinger, "Interaction-aware motion prediction for autonomous driving: A multiple model kalman filtering scheme," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 80–87, 2021.

[42] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? a unified framework for maneuver classification and motion prediction," *IEEE Transactions on Intelligent Vehicles*, pp. 129–140, 2018.

[43] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in 2018 *IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1179–1184.

[44] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," *Advances in Neural Information Processing Systems*, vol. 32, pp. 15 424–15 434, 2019.

[45] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018.

[46] K. Messaoud, I. Yahiaoui, A. Verroust-Blondet, and F. Nashashibi, "Attention based vehicle trajectory prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 175–185, 2020.

[47] Z. Zhao, H. Fang, Z. Jin, and Q. Qiu, "Gisnet: Graph-based information sharing network for vehicle trajectory prediction," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[48] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[49] Y. Wang, S. Zhao, R. Zhang, X. Cheng, and L. Yang, "Multi-vehicle collaborative learning for trajectory prediction with spatio-temporal tensor fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–13, 2020.

[50] K. Saleh, M. Hossny, and S. Nahavandi, “Long-term recurrent predic tive model for intent prediction of pedestrians via inverse reinforcement learning,” in *2018 Digital Image Computing: Techniques and Applications (DICTA). IEEE*, 2018, pp. 1–8.

[51] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, “Imitating driver behavior with generative adversarial networks,” in *2017 IEEE Intelligent Vehicles Symposium (IV). IEEE*, 2017, pp. 204–211.

[52] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, “Large-scale cost function learning for path planning using deep inverse reinforcement learning,” *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[53] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, “Neighbourhood context embeddings in deep inverse reinforcement learning for predicting pedestrian motion over long time horizons,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[54] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan et al., “Argoverse: 3d tracking and forecasting with rich maps,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.

[55] J. Mercat, T. Gilles, N. El Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, “Multi-head attention for multi-modal joint vehicle motion forecasting,” in *2020 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2020, pp. 9638–9644.

[56] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal et al., “Scene transformer: A unified multi-task model for behavior prediction and planning,” *International Conference on Learning Representations (ICLR)*, 2021.

[57] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, “Multimodal motion prediction with stacked transformers,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7577–7586.

[58] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, “Learning lane graph representations for motion forecasting,” in *European Conference on Computer Vision.* Springer, 2020, pp. 541–556.

[59] J. Gu, C. Sun, and H. Zhao, “Densetnt: End-to-end trajectory prediction from dense goal sets,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 303–15 312.

[60] W. Zeng, M. Liang, R. Liao, and R. Urtasun, “Lanercnn: Distributed representations for

graph-centric motion forecasting," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2021, pp. 532–539.

[61] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, "Learning to predict vehicle trajectories with model-based planning," *arXiv preprint arXiv*:2103.04027, 2021.

[62] A. Geiger, P. Lenz, C. Stiller, R. Urtasun. "The KITTI Vision Benchmark Suite", cvlibs.net/datasets/kitti/ (Accessed: 20 April 2023).

[63] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G. and Beijbom, O., nuScenes: A multimodal dataset for autonomous driving, *CVPR*, (2020), 11618-11628, doi: 10.1109/CVPR42600.2020.01164.

[64] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D. and Hays, J., Argoverse: 3D Tracking and Forecasting With Rich Maps, *IEEE Conference on Computer Vision and Pattern Recognition,* (2019), 8748-8757, doi: 10.1109/CVPR.2019.00895.

[65] Coifman, B. A critical evaluation of the next generation simulation (NGSIM) vehicle trajectory dataset, *Trans. Res. B Methodol*., 105 (2017), 362-377, doi: 10.1016/j.trb.2017.09.018.

[66] Caesar, H., Kabzan, J., Tan, K., nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles, *CVPR ADP3 workshop*, (2021), doi: 10.48550/arXiv.2106.11810

[67] Caesar, H., Kabzan, J., Tan, K. nuImages. "Nuscenes.org", nuscenes.org/nuimages (Accessed: 20 April 2023).

[68] Batz, K. Watson, and J. Beyerer, "Recognition of dangerous situations within a cooperative group of vehicles," in *Intelligent Vehicles Symposium*, 2009.